UNLIMITED DISTRIBUTION 85-04076

National Defence
Research and Development Branch

Défense Nationale
Bureau de Recherche et Développment

FIVPRE:
A PRE-PROCESSOR FOR THE CONCEPT EXPLORATION MODEL SHOP5

Bonny Lee
J.L. Colwell P.V. Godreau

AUGUST 1985

Approved by L.J. Leggat H/Hydronautics Section

DISTRIBUTION APPROVED BY J. Ozanett

D/TD

TECHNICAL COMMUNICATION 85/310

Defence Research Establishment Atlantic

Centre de Recherches pour la Défense Atlantique

Canada

ABSTRACT

FIVPRE is an interactive pre-processor for SHOP5, the DREA Concept Exploration Model for conventional monohull frigates and destroyers. FIVPRE provides the user with a simple means to create and modify the input files for SHOP5. Prior knowledge of the format of these files is not required. Through a simple command language, FIVPRE can be used to define new or change existing values of SHOP5 input parameters from the terminal. FIVPRE also keeps records of the files it has created. This Technical Communication is a user's manual for FIVPRE, describing the FIVPRE commands and giving examples of terminal sessions. The appendices contain a brief description of SHOP5 input and program documentation of FIVPRE, including descriptions of all the procedures in FIVPRE.

RESUME

Le FIVPRE est un préprocesseur interactif du SHOP5, le modèle d'exploration des concepts qu'utilise le CRDA pour les frégates et les destroyers monocoques classiques. Le FIVPRE fournit à l'usager un moyen facile de créer les fichiers d'entrée pour le SHOP5 ou de modifier ces fichiers, et cela sans avoir à connaître d'avance le format des fichiers en question. A partir du terminal et au moyen d'un langage de commande simple, on peut se servir du FIVPRE pour définir de nouvelles valeurs dans les paramètres d'entrée du SHOP5 ou pour modifier les valeurs existantes. Le FIVPRE répertorie également les fichiers qu'il a créés. La présente communication technique est un guide de l'usager du FIVPRE qui décrit les commandes du système et donne des exemples de séances d'utilisation au terminal. Les annexes contiennent une brève description des entrées SHOP5 et des renseignements sur la programmation du FIVPRE comprenant notamment des descriptions de toutes les procédures applicables au préprocesseur.

TABLE OF CONTENTS

	ABSTRACT	ii
1.	INTRODUCTION	1
	1.1 Purpose of FIVPRE	
	1.2 An Overview of SHOP5 Input Variables	
2.	FEATURES OF FIVPRE	2
	2.1 The FIVPRE Command Language	
	2.2 User-Friendly Features	
3.	COMMANDS IN FIVPRE	5
	3.1 CHANGE	
	3.2 DEFAULT	
	3.3 DISPLAY	
	3.4 DUMP	
	3.5 ENTER	
	3.6 EXIT	
	3.7 FIND	
	3.8 HELP	
	3.9 NEW	
4.	TWO EXAMPLES OF USING FIVPRE	9
	4.1 Creating a New File with FIVPRE	
	4.2 Editing an Existing File with FIVPRE	
5.	CONCLUDING REMARKS	10
	TABLE	10
	APPENDICES	11
	A. DESCRIPTION OF SHOP5 INPUT VARIABLES	
	B. CREATING A NEW FILE (SAMPLE TERMINAL SESSION)	
	C. EDITING AN EXISTING FILE (SAMPLE TERMINAL SESSION)	
	D. MACHINE DEPENDENCIES	
	E. GENERAL PROGRAM STRUCTURE OF FIVPRE	
	F. COMMAND PROCESSING SUBROUTINES	
	G. THE FIVPRE DICTIONARIES AND ASSOCIATED SUBROUTINES	
	H. COMMAND EXECUTION SUBROUTINES	
	I. UTILITY SUBROUTINES	
	REFERENCES	70

1. INTRODUCTION

A "concept exploration model" is a simplified form of a ship synthesis model which addresses the earliest phase of the ship selection process. It is a computer program which takes as input a set of ship dimensions and simple operational requirements, and calculates therefrom ship stability, performance, and other capabilities. Sufficient constraints on ship geometry and stability are incorporated into the program to ensure that unrealistic configurations are rejected; hence performance is estimated only for ships which are feasible from an engineering point of view.

1.1 Purpose of FIVPRE

FIVPRE is an interactive pre-processor for SHOP5 (1), the DREA Concept Exploration Model for conventional monohull frigates and destroyers. SHOP5 uses a data file which must have a particular format. With FIVPRE, the user can create and modify data files to be used as SHOP5 input, without having to know the exact format of the input file.

1.2 An Overview of SHOP5 Input Variables

SHOP5 input includes parameters which describe a ship or number of ships, and parameters which control the execution of the program. All these parameters, or 'variables', are organized into sets of variables, or 'records'. There are 11 records altogether, but not all of them are used in every set of input. Their titles are as follows:

Record 1: Title
Record 2: Program Control Integers
Record 3: Primary Input
Record 4: Re-definition of Method Control Integers
Record 5: Re-definition of Optional Input
Record 6: Re-definition of Rejection and Seakeeping Criteria
Record 7: User-supplied Appendage Resistance Coefficient
Record 8: User-supplied Overall Propulsive Coefficient
Record 9: Engine Algorithm Data
Record 10: User-supplied Generator Data
Record 11: User-supplied Cost Factors

Records 1,2, and 3 are required input. Record 1 contains only one variable, which is the title of the file. Record 2 contains 8 integers which control the execution of SHOP5. Record 3 contains variables which describe the hull form and which specify the operational requirements for a ship or set of ships.

Records 4,5, and 6 supply additional information for SHOP5. Record 4 contains up to 9 integers, which determine the design methods used by SHOP5. The 9 method control integers have default values, but any or all of the integers may be re-defined by the user. Record 5 contains the optional input variables which describe the ship or set of ships in more detail. There are 44 optional input variables, and they all have default values. Like the method control integers, these optional input variables may be re-defined by the user. Record 6 contains the rejection and seakeeping criteria. There are 7 rejection criteria and 4 seakeeping criteria. Like the variables of Record 4 and Record 5, they have default values that can be re-defined by the user. For Records 4, 5, and 6, only the variables which are re-defined by the user appear in the input data file for SHOP5.

Records 7,8,9,10, and 11 are not required unless certain method control integers are re-defined by the user. Record 7 contains the user-supplied appendage resistance coefficient. Record 8 contains the user-supplied overall propulsive coefficient. Record 9 contains the engine algorithm data. Record 10 contains the user-supplied generator data. Record 11 contains the user-supplied cost factors. Record 11 has a similar format to that of Records 4,5,and 6, and contains only those cost factors out of a possible 16 which have been re-defined by the user.

For more information on the SHOP5 input variables, see Appendix A or the SHOP5 User's Manual (1).

2. FEATURES OF FIVPRE

The program FIVPRE acts as an intermediate storage area for SHOP5 input variables. Using FIVPRE, the user can define the values for input variables, either from the terminal or from an existing data file; he can display these variables and change their values; and he can write them on a data file which will be the input for SHOP5.

2.1 The FIVPRE Command Language

Through the FIVPRE command language, the user can tell FIVPRE what action he wants to take, and which variable the action should be applied to. Thus, a FIVPRE instruction has the syntax

<action> <specifier>

where <action> is the command word and <specifier> is a variable name or a record name. For some commands, <specifier> is not needed.

The algorithms used by FIVPRE to recognize commands are based on algorithms developed by Hally and Dent (2). FIVPRE uses a dictionary of command words to find which command the user wants. Since a command word may be uniquely specified by its first few letters, the user need not type in the whole command word. For example, the command "CHANGE" may be specified by "C", since "CHANGE" is the only word in the dictionary which starts with "C". On the other hand, the commands "DEFAULT", "DISPLAY", and "DUMP" all start with "D", so "D" would not uniquely identify a command. However, "DE" would be recognized as "DEFAULT" by the dictionary, since no other command word starts with "DE". Similarly, "DI" would be recognized as "DISPLAY", and "DU" would be recognized as "DUMP".

FIVPRE uses two dictionaries to find which variable or record the user wants: one dictionary uses the variable's "official" abbreviation, while the other uses its description (enough to uniquely identify the variable). FIVPRE first looks through its dictionary of abbreviations. Then, if it can't find an abbreviation that matches the word specified by the user, FIVPRE looks through its dictionary of descriptions for phrases containing the words specified by the user.

The abbreviation dictionary and description dictionary contain the abbreviations and descriptions of records as well as variables; therefore, the user can specify records in much the same way as he specifies variables. An example of a command which refers to a record is: "DISPLAY PRIMARY INPUT". The command "DISPLAY RECORD 3" would have the same effect. Table 1 contains the abbreviations and descriptions of the records.

An example of how FIVPRE would identify a variable follows. Suppose the user wants to display the superstructure length, which has the abbreviation LS in the User's Manual for SHOP5. The user could specify this variable by using its abbreviation LS, by using all of its description, SUPERSTRUCTURE LENGTH, or by using only part of its description: SUPER LEN. Suppose the user types the command "DISPLAY SUPER LEN". To identify the variable, FIVPRE looks through its abbreviation dictionary for "SUPER". It doesn't find a matching entry in the dictionary; it then looks through its description dictionary for any description which contains both "SUPER" and "LEN". In this example, LS, the superstructure length, would be the only matching description.

Sometimes the description specified by the user could apply to more than one variable. In this situation, FIVPRE will list the possible matching variable descriptions and ask the user to pick one. For example, suppose the user types "DISPLAY SU LE". FIVPRE responds with:

```
** AMBIGUOUS INPUT **

1  Ls:   SUPERSTRUCTURE LENGTH
2  AK1:  COST FACTOR: HULL (LESS SUPERSTRUCTURE)
3  AK15: COST FACTOR: MISC. SURCHARGES, LEAD SHIP

ENTER SELECTION NUMBER (OR 99 FOR EXIT) =>
```

The user would type in 1, 2, 3, or 99, depending on which variable he wanted.

If there are more than 15 possible choices, FIVPRE will not display them. Instead, it will print out a message saying that the record or variable name is too ambiguous.

2.2 User-Friendly Features

Several FIVPRE features make the program easier to use. These features include helpful commands, extensive use of prompts when changing the values of variables, checking routines for validating input from the user, and special input values which can be used in response to prompts. In addition, FIVPRE will write out a brief introduction to the program at the beginning of its run, if the user desires.

Helpful FIVPRE commands are NEW, HELP, and DEFAULT. All three commands are described in detail in Section 3.

The NEW command is particularly useful for the user who is not familiar with the SHOP5 input variables. This command prompts the user for all the data needed to create a SHOP5 input file; the user is only required to respond to the prompts.

The HELP command gives information about FIVPRE commands. It either displays a list of all the FIVPRE commands or displays information about a specific command. This is helpful for the user who is not familiar with the FIVPRE command language.

The DEFAULT command displays the default value of a specified variable. For example, the command "DEFAULT VS" would display the default value of VS, the superstructure volume. This command is useful when the user is trying to decide whether the default value of a variable is appropriate or whether he should re-define the value of the variable.

FIVPRE uses prompts to request input when the user is changing the value of a variable through the keyboard. There are two types of prompts: direct prompts and menu prompts. A direct prompt is an open-ended request for data, e.g. ENTER THE VALUE FOR Ls: SUPERSTRUCTURE LENGTH => . A menu prompt asks the user to make a selection from a list of options, e.g.

TYPE OF ENGINES

0 - SHOP5 RUBBER ENGINES
1 - USER SUPPLIED RUBBER ENGINES
2 - SHOP5 REAL ENGINES
3 - USER SUPPLIED REAL ENGINES

ENTER NUMBER = >

The user then types in the number of his choice. This type of prompt is used when a variable has a limited number of allowed values.

Invalid input in response to a prompt results in an error message and another prompt. "Invalid input" means that non-numeric characters are used when a number is required (except for the special input values noted below), or an input value is outside the allowed range in a menu prompt. For example, if the prompt is "ENTER THE VALUE FOR Ls: SUPER-STRUCTURE LENGTH = >", and the user types a number using the letter O instead of the digit 0, a message will be displayed: "**INVALID INPUT**". The program will then prompt the user again. This gives the user another chance if he should hit the wrong key when tvping a response to a prompt.

Two non-numeric characters which are not invalid input are the character "*" and the word "DEF". They have special functions in FIVPRE. When data is requested through a direct prompt, the user may use the character "*", to mean "leave the value of variable the way it is". For example, suppose the user types "CHANGE LS" and then decides against changing LS (the superstructure length); when the prompt appears, the user can type "*" and the value of LS will not be changed.

The other special input value, "DEF", means "default value" when typed in reponse to a direct prompt. For example, suppose the prompt is "ENTER VALUE FOR Vs: SUPERSTRUCTURE VOLUME >". Typing "DEF" will set VS to the default value. In effect, "DEF" cancels the re-definition of the variable. Note that this response can only be used for optional input variables, rejection and seakeeping criteria, and cost factors. Using "DEF" for other variables will result in an error message. Note: The command DEFAULT should not be confused with the special input value "DEF"; DEFAULT displays the default value, while "DEF" sets the default value.

3. COMMANDS IN FIVPRE

3.1 CHANGE

The CHANGE command changes the values of a set of variables (a record) or a single variable. The format for the command is seen below:

CHANGE variable name Example: CHANGE TITLE
CHANGE record name Example: CHANGE REC4

After a CHANGE command is entered, a prompt or series of prompts will appear. The type of prompt depends on which variable is being changed; some variables use a menu prompt, others use a direct prompt. If an entire record is being changed, FIVPRE will generate prompts for all the variables in the record. For records 4,5,and 6, a menu prompt is used for the entire record, giving the user a choice of variables to re-define.

In some cases, the command to change a single variable will result in a series of prompts rather than a single prompt. This can occur in two situations. In one situation, changing the value of one variable may affect several other variables, which would need to be re-defined. For example, changing the mode from search to describe would require that the primary input variables be re-defined. In the other case, the variable specified in the command belongs to a subgroup of variables. An example of such a subgroup is CPLO, CPHI, and I3 (minimum prismatic coefficient, maximum prismatic coefficient, and number of prismatic coefficients to be considered). If the user wants to change only one parameter in the subgroup, he can use the special character "*" in response to the additional prompts.

3.2 DEFAULT

The DEFAULT command displays the default value of a variable. The format for the command is seen below:

DEFAULT variable name Example: DEFAULT NDECK

After the DEFAULT command is entered, FIVPRE displays the default value of the variable. If the variable does not have a default value, FIVPRE will print a message telling the user that there is no default value for this variable. Note that it is not possible to use this command to display the default values of a record; this command only works for single variables.

3.3 DISPLAY

The DISPLAY command displays the value of a single variable, the values of a record, or the values of all the variables. The format for the command is seen below:

DISPLAY variable name Example: DISPLAY MODE
DISPLAY record name Example: DISPLAY RECORD 3
DISPLAY ALL

For most records and variables, the format used to display the values of the variables is straightforward. However, for a few of the records, some explanation may be needed. For Record 4, the re-definition of the method control integers, all the method control integers are displayed, whether they have been re-defined or not. The integers that have been re-defined are marked with a "*" in the left-hand margin. For Records 5 and 6, only the re-defined variables are displayed, in order to improve the readability of the display.

3.4 DUMP

The DUMP command writes the values of all the variables onto a file called FIVIN, which is the input file for SHOP5. The user may also specify the name of another file in this command, in which case FIVPRE writes the data both to this file and to FIVIN. The format for the command is seen below:

```
DUMP
DUMP file name          Example:  DUMP SAMPLE
```

If the user has not specified a file name in the DUMP command, FIVPRE will ask the user for the name of the file in which he wants the input data stored. If the the user hits the return key without typing a file name, FIVPRE will write the data to the file FIVIN.

If the specified file is not FIVIN, FIVPRE checks to see if the file already exists. If this is the case, FIVPRE will tell the user and ask the user if he wants to replace the existing file (thus destroying whatever was previously written on the file). If the user doesn't want to replace the file, he will be asked for another file name.

After the user has either entered the name of a new file or decided to replace an existing file, FIVPRE writes the data to the specified file. The data is also written to the file FIVIN.

Note that data is not really saved unless the user specifies a file other than FIVIN, since FIVIN is always overwritten when a DUMP is done.

3.5 ENTER

The ENTER command enters values of variables from an existing data file. The format for the command is seen below:

```
ENTER
ENTER file name         Example:  ENTER FIVIN
```

If the user has not specified a file name in the ENTER command, FIVPRE will ask the user which file he wants to enter. If the file exists, FIVPRE reads the data from the file into the appropriate variables. If the file does not exist, FIVPRE will ask the user if he wants to specify another file.

3.6 EXIT

The EXIT command stops the program. The format for this command is seen below:

EXIT

When the EXIT command is entered, the user exits from FIVPRE. If the user has changed the values of some variables and has not saved the values (through the DUMP command), FIVPRE will ask if the user wants to save the current file.

3.7 FIND

The FIND command searches through a library file, using a search key supplied by the user, and prints out the names of data files which contain the search key in their titles. The format for the command is seen below:

FIND

After the FIND command is entered, FIVPRE asks the user for a search key. This is a word, phrase, or string of letters which FIVPRE looks for in the title of any existing data file. For example, if the user wants FIVPRE to list any files which have "HELICOPTER" in their titles, then "HELICOPTER" is the search key. As FIVPRE looks through the library file, it prints out the title and the file name of any files which contain the search key in their title. If the user hits the return key without typing in a search key, FIVPRE will list all the entries in the library file.

The library file is called FIVLIB and contains a list of the titles and file names of data files which have been created by FIVPRE. FIVPRE updates FIVLIB by several different methods. At the beginning of each session, FIVPRE checks through FIVLIB and deletes those entries whose corresponding files do not exist on the system; this ensures that files which have been removed from the system since the last use of FIVPRE are no longer listed in FIVLIB. Each time the DUMP command is used, the title and file name are added to FIVLIB. If a user enters data from an existing file which is not listed in FIVLIB, FIVPRE will add the title and file name of the file to FIVLIB. No more than 30 files can be listed in FIVLIB; any files created after FIVLIB is full will not be listed in FIVLIB.

3.8 HELP

The HELP command lists all the FIVPRE commands and also gives information about a particular command. The format for this command is seen below:

```
HELP
HELP command              Example:  HELP DISPLAY
```

"HELP" lists all the commands in FIVPRE, while "HELP command" tells the user the purpose and format of the specified command. The command "HELP SPECIAL" will print an explanation of the two special input symbols, "*" and "DEF".

3.9 NEW

The NEW command changes the values of all the variables. It is used when creating new data files. The format for the command is seen below:

```
NEW
```

If the user has made changes to the values of any variables and has not done a DUMP before entering the command NEW, he will be asked whether he wants to continue with the NEW command, since the existing values of all the variables will be erased. If the user wishes to continue, FIVPRE will proceed with the execution of the NEW command.

The command NEW generates a series of prompts for all the variables needed to create a new file. These variables are not automatically written to a file, hence the user has a chance to edit the data before using the DUMP command to write the data to a file.

4. TWO EXAMPLES OF USING FIVPRE

4.1 Creating A New File With FIVPRE

To create a new file using FIVPRE, only the commands NEW and DUMP are needed. Appendix B contains a sample terminal session using FIVPRE to create a file. The commands used in the sample terminal session are NEW and DUMP.

4.2 Editing An Existing File With FIVPRE

Appendix C contains a sample terminal session using FIVPRE to edit an existing file. The commands used in this session are HELP, FIND, ENTER, CHANGE, DISPLAY, and DUMP.

5. CONCLUDING REMARKS

FIVPRE provides a friendly interface between the user and a SHOP5 input file. With FIVPRE, new data files can be created and modified without the exact knowledge of the format of the SHOP5 input or the computer system editor. FIVPRE also keeps track of the files it creates, and so the user does not have to remember the names of all the SHOP5 input files he has created. In general, FIVPRE makes the use of SHOP5 much easier.

TABLE 1: RECORD NAMES

Record Number	Abbreviation	Description
1	REC1	TITLE
2	REC2	PROGRAM CONTROL INTEGERS (PCI)
3	REC3	PRIMARY INPUT
4	REC4	METHOD CONTROL INTEGERS (MCI)
5	REC5	OPTIONAL INPUT
6	REC6	REJECTION/SEAKEEPING CRITERIA
7	REC7	USER-SUPPLIED APPENDAGE RESISTANCE COEFFICIENT
8	REC8	USER-SUPPLIED OVERALL PROPULSIVE COEFFICIENT
9	REC9	ENGINE ALGORITHM DATA
10	REC10	USER-SUPPLIED GENERATOR DATA
11	REC11	USER-SUPPLIED COST FACTORS

APPENDIX A

DESCRIPTION OF SHOP5 INPUT VARIABLES

(Adapted from SHOP5 User's Manual)

For more detailed descriptions of the SHOP5 input variables, see the SHOP5 User's Manual.

RECORD (1): TITLE

TITLE — Alphanumeric title shown on output, Maximum 80 characters. This title is written on lineprinter output from both SHOP5 and the post-processor.

RECORD (2): PROGRAM CONTROL INTEGERS

MODE — Determines which of SEARCH or DESCRIBE modes is to be used.

INOUT — Controls the system of units for input and output.

IPOST — Controls the use of disk files for storage of output data for the post-processor, FIVPOS.

ILPT — Controls the use of lineprinter for output of ship data from the SEARCH mode; must be input for both SEARCH and DESCRIBE mode.

ILEGND — Controls the output of a legend of all parameters used in normal lineprinter output; legend includes parameter name, description, and units.

NCON — The number of Method Control Integers to be re-defined; default method will be used only for those design method options not re-defined. See Record (4).

NOPTN — The number of Optional Input to be re-defined; default values will be used only for those Optional Input not re-defined. See Record (5).

NLIM — The number of rejection and seakeeping criteria to be re-defined; default values will be used only for those parameters not re-defined. See Record (6).

RECORD (3): PRIMARY INPUT

This record provides the required input for definition of hull form and operational requirements. Records (3.1) and (3.2) are for the SEARCH and DESCRIBE modes, respectively.

(3.1): PRIMARY INPUT, SEARCH MODE
Program Control Integer MODE = 0

Independent Variables (hull form)

XXXLO	Minimum Characteristic dimension value.
XXXHI	Maximum Characteristic dimension value.
I1	Number of Characteristic dimension values to be considered. See Note 1.
CRMLO	Minimum length/displacement ratio.
CRMHI	Maximum length/displacement ratio.
I2	Number of length/displacement ratios to be considered.
CPLO	Minimum prismatic coefficient.
CPHI	Maximum prismatic coefficient.
I3	Number of prismatic coefficients to be considered.
CBLO	Minimum block coefficient.
CBHI	Maximum block coefficient.
I4	Number of block coefficients to be considered.
BOTLO	Minimum beam/draft ratio.
BOTHI	Maximum beam/draft ratio.
I5	Number of beam/draft ratios to be considered.

Operational Requirements

E	Minimum acceptable range at endurance speed (miles).
WC	Combat system weight (ton;tonne).
VD	Design speed (knots).
VC	Cruise speed (knots).
VE	Endurance speed (knots).
VW	Seakeeping speed (knots).
HW	Seakeeping significant wave height (ft;m).

Notes: (1) The type of input expected for the characteristic dimension depends on the value of Method Control Integer IDIMEN. See Record (4).

(2) I1 x I2 x I3 x I4 x I5 must be < 801.

(3) Each of I1, I2, I3, I4, I5 must be < 12.

(3.2): PRIMARY INPUT, DESCRIBE MODE
Program Control Integer MODE > 0.

Independent Variables (hull form)

XXX	Characteristic dimension, see Note 1 for Record (3.1).
CRM	Length/displacement ratio (Froude notation, Circle M).
CP	Prismatic coefficient.
CB	Block coefficient.
BOT	Beam/draft ratio.

Operational Requirements

WC	Combat system weight (ton;tonne).
VD	Design speed (knots).
VC	Cruise speed (knots).
VE	Endurance speed (knots).
VW	Seakeeping speed (knots).
HW	Seakeeping significant wave height (ft;m).

Notes: (1) A maximum of 11 ships may be considered by the DESCRIBE mode.

RECORD (4): Re-definition of METHOD CONTROL INTEGERS
Only used if Program Control Integer NCON > 0.

This record contains the numbers and new values of the Method Control Integers which have been re-defined.

Descriptions of Method Control Integers:

NUMBER	NAME	VALUE	DESCRIPTION
1	IDIMEN	0	Characteristic dimension = displacement
		1	Characteristic dimension = length
		2	Characteristic dimension = beam
		3	Characteristic dimension = draft
2	IRESID	0	Residuary resistance from NRC FSS data
		1	Residuary resistance from Taylor, Hamburg C, or NPL Series data
3	IAPPND	0	SHOP5 Appendage resistance calculations
		1	User-supplied appendage resistance coefficient
4	IPROP	0	SHOP5 OPC calculations, C. P. propeller
		1	SHOP5 OPC calculations, fixed pitch propeller
		2	User-supplied overall propulsive coefficients

NUMBER	NAME	VALUE	DESCRIPTION
5	ISTRUC	0	Homogeneous hull and superstructure
		1	Hybrid: Steel hull and aluminum superstructure
6	IENGIN	0	SHOP5 rubber engines
		1	User-supplied rubber engines
		2	SHOP5 hierarchy and SHOP5 data base
		3	User-supplied hierarchy, SHOP5 data base
		4	SHOP5 hierarchy, user-supplied data base
		5	User-supplied hierarchy and data base
7	ICHOOS	0	SEARCH: must satisfy input engine configuration DESCRIBE: may use engine hierarchy
		1	SEARCH: may use engine hierarchy DESCRIBE: must satisfy input engine configuration
8	IGEN	0	SHOP5 gas turbine generators
		1	SHOP5 diesel generators
		2	User-supplied generator characteristics
9	ICOST	0	Standard cost factors (1977 dollars)
		1	User-supplied cost factors

Note: (1) Re-definition of Method Control Integers may require additional user-supplied input as follows:

Number	Name	Required Input
3	IAPPND	Record (7)
4	IPROP	Record (8)
6	IENGIN	Record (9)
8	IGEN	Record (10)
9	ICOST	Record (11)

RECORD (5): Re-definition of OPTIONAL INPUT
Only used if Program Control Integer NOPTN > 0.

This record contains the numbers and new values of the Optional Input which have been re-defined.

Descriptions of Optional Input

NUMBER	NAME	DESCRIPTION
1	FM	MIDSHIPS FREEBOARD, DESCRIBE MODE
2	CW	WATERPLANE COEFFICIENT
3	LCB	LONGITUDINAL CENTRE OF BUOYANCY
4	LCF	LONGITUDINAL CENTRE OF FLOTATION
5	NP	COMPLEMENT
6	NDECK	NUMBER OF INTERNAL DECKS
7	NBULK	NUMBER OF WATERTIGHT COMPARTMENTS
8	C	COMPARTMENT STANDARD OF FLOODING
9	LS	SUPERSTRUCTURE LENGTH
10	FFP	FREEBOARD AT FORWARD PERPENDICULAR
11	TOS	WAVE MODAL PERIOD FOR HW (SEAKEEPING)
12	AMAX	VERTICAL ACCELERATION/G, 0% CREW EFFECTIVENESS
13	AMIN	VERTICAL ACCELERATION/G, 100% CREW EFFECTIVENESS
14	CITTC	ITTC CORRELATION ALLOWANCE
15	DEHP	DESIGN MARGIN ON EFFECTIVE POWER EHP=(1+DEHP)R*V
16	DIA	PROPELLER DIAMETER
17	NSH	NUMBER OF PROPELLER SHAFTS
18	NGEN	NUMBER OF ELECTRICAL GENERATORS
19	PG	ELECTRICAL POWER INSTALLED
20	PGE	AVERAGE CRUISE ELECTRICAL POWER
21	HWR	SIGNIFICANT WAVE HEIGHT FOR RANGE CALCULATIONS
22	TOR	WAVE MODAL PERIOD FOR HWR
23	YIELD	YIELD STRENGTH OF HULL MATERIAL
24	DENS	DENSITY OF HULL MATERIAL
25	FICE	FACTOR FOR ICE STRENGTHENING
26	RAFT	MARGIN: MAIN ENGINE RAFTING
27	GEAR	MARGIN: GEAR BOX WEIGHT
28	DG1	MARGIN: GROUP 1 WEIGHT (HULL STRUCTURE)
29	DG2	MARGIN: GROUP 2 WEIGHT (PROPULSION MACHINERY)
30	DG3	MARGIN: GROUP 3 WEIGHT (ELECTRICAL)
31	DG5	MARGIN: GROUP 5 WEIGHT (AUXILIARIES)
32	DG6	MARGIN: GROUP 6 WEIGHT (OUTFIT AND FURNISHING)
33	DWB	MARGIN: BASIC WEIGHT
34	DWD	MARGIN: DISPOSABLE WEIGHT
35	KGX	VCG OF EXTRA BASIC WEIGHT
36	KGC	VCG OF COMBAT SYSTEM WEIGHT
37	VS	SUPERSTRUCTURE VOLUME
38	DVM	MARGIN: MACHINERY VOLUME
39	DVO	MARGIN: SHIP SYSTEMS AND OUTFIT VOLUME
40	DVB	MARGIN: BASIC VOLUME
41	DVN	MARGIN: PERSONNEL VOLUME
42	TD/T	NORMALIZED TIME AT DESIGN SPEED
43	TC/T	NORMALIZED TIME AT CRUISE SPEED
44	TE/T	NORMALIZED TIME AT ENDURANCE SPEED

Note: (1) Optional Input may be re-defined as positive or negative numbers, indicating, respectively, whether the new value is an absolute quantity or a ratio.

RECORD (6): Re-definition of REJECTION AND SEAKEEPING CRITERIA
Only used if Program Control Integer NLIM > 0.

This record contains the numbers and new values of the rejection criteria (SEARCH mode) or of the seakeeping criteria (DESCRIBE mode) which have been re-defined.

(6.1): Re-definition of REJECTION CRITERIA, SEARCH mode only
Program Control Integers MODE = 0 and NLIM > 0.

Description of Rejection Criteria

NUMBER	NAME	DESCRIPTION	RE-DEFINABLE
1	N WET	Number of deck wetnesses/hour	yes
2	ACCN 4	Vertical acceleration at Station 4/G	yes
3	SLAM F	Slam force / displacement	yes
4	S EFF	Seakeeping effectiveness	yes
5	F MID	Midship freeboard	no
6	C MID	Midship coefficient	no
7	PROP E	Overall propulsive coefficient at endurance speed	yes
8	ENGINE	Engine configuration	no
9	RANGE	Range at endurance speed	no
10	KG MAX	Vertical centre of gravity	yes
11	VOLUME	Combat system volume	yes

Note: (1) Only 7 of the 11 rejection criteria may be re-defined, as shown above.

(6.2): Re-definition of SEAKEEPING CRITERIA, DESCRIBE mode only
Program Control Integers MODE > 0, NLIM > 0.

The seakeeping criteria are identical to the first four rejection criteria, as shown above.

RECORD (7): User-supplied APPENDAGE RESISTANCE COEFFICIENT
Only used if Method Control Integer IAPPND = 1.

CAPP Appendage resistance coefficient.

Note: (1) The appendage resistance is equal to the total ship frictional resistance multiplied by CAPP.

RECORD (8): User-supplied OVERALL PROPULSIVE COEFFICIENTS
Only used if Method Control Integer IPROP = 2.

ETAD Overall propulsive coefficient at design speed.
ETAC Overall propulsive coefficient at cruise speed.
ETAE Overall propulsive coefficient at endurance speed.

RECORD (9): ENGINE ALGORITHM DATA
Only used if Method Control Integer IENGIN > 0.

(9.1): User-supplied RUBBER ENGINE DATA
Only used when Method Control Integer IENGIN = 1.

SPW2 Specific weight of propulsion system (lb/HP; kg/kW).
SFCR Specific fuel consumption of engines, at full power (lb/HP-Hr; kg/kW-Hr).

(9.2): SHOP5 Hierarchy and SHOP5 Engine Data Base
Only used when Method Control Integer IENGIN = 2.

ITYPE Engine configuration per shaft, one of the values below.
= 1, Single gas turbine
= 2, COGAG
= 3, COGOG
= 4, CODAG
= 5, CODOG
= 6, CODAD

IFUTUR Controls which engine data base is to be used (1980 engine data base or 1995 engine data base)

IHIAR Controls which hierarchy is to be used for engine choice, according to the primary consideration (noise generation, gearbox complexity, or progressive power)

(9.3): User-supplied Hierarchy, SHOP5 Engine Data Base
Only used when Method Control Integer IENGIN = 3.

ITYPE	Engine configuration per shaft (see Record (9.2))
IFUTUR	Controls which engine data base is to be used (1980 engine data base or 1995 engine data base)
HIARK(I)	I=1,6 Input of hierarchy, six numbers corresponding to those shown for ITYPE in Record (9.2)

(9.4): SHOP5 Hierarchy, User-supplied Engine Data Base
Only used when Method Control Integer IENGIN = 4.

ITYPE	Engine configuration per shaft (see Record (9.2))
IHIAR	Controls which hierarchy is to be used for engine choice, according to the primary consideration (noise generation, gearbox complexity, or progressive power)
SPW2B	Specific weight of propulsion system, not including engines (lb/HP; kg/kW)
NTURB	Number of gas turbine engines to be input by user.
NAMTUR(I)	Name of turbine(I), maximum 16 characters.
POWT(I)	Maximum power of turbine(I) (HP; kW).
SFCT(I)	Specific fuel consumption of turbine(I) (lb/HP-Hr; kg/kW-Hr).
WTT(I)	Weight of turbine(I) (ton;tonne).
NDIES	Number of diesel engines to be input by user.
NAMDIE(I)	Name of diesel(I), maximum 16 characters.
POWD(I)	Maximum power of diesel(I) (HP; kW).
SFCD(I)	Specific fuel consumption of diesel(I) (lb/HP-Hr; kg/kW-Hr).
WTD(I)	Weight of diesel(I) (ton;tonne).

(9.5): User-supplied Hierarchy and Engine Data Base
Only used when Method Control Integer IENGIN = 5.

ITYPE	Engine configuration per shaft (see Record (9.2))
SPW2B	Specific weight of propulsion system, not including engines (lb/HP; kg/kW)
HIARK(I)	I=1,6 Input of hierarchy, six numbers corresponding to those shown for ITYPE in Record (9.2)
NTURB	Number of gas turbine engines to be input by user.
NAMTUR(I)	Name of turbine(I), maximum 16 characters.
POWT(I)	Maximum power of turbine(I) (HP; kW).
SFCT(I)	Specific fuel consumption of turbine(I) (lb/HP-Hr; kg/kW-Hr).
WTT(I)	Weight of turbine(I) (ton;tonne).

NDIES Number of diesel engines to be input by user.
NAMDIE(I) Name of diesel(I), maximum 16 characters.
POWD(I) Maximum power of diesel(I) (HP; kW).
SFCD(I) Specific fuel consumption of diesel(I) (lb/HP-Hr; kg/kW-Hr).
WTD(I) Weight of diesel(I) (ton;tonne).

RECORD (10): User-supplied GENERATOR DATA

Only used when Method Control Integer IGEN = 2.

SFCG Specific fuel consumption of generator set (lb.HP-Hr; kg/kW-Hr)
SPGEN Specific weight of electrical system (lb/kW; kg/kW)

RECORD (11): User-supplied COST FACTORS

Only used when Method Control Integer ICOST = 1.

This record contains the numbers and new values of the Optional Input which have been re-defined.

Description of Cost Factors

NUMBER	NAME	DESCRIPTION
1	AK1	HULL (LESS SUPERSTRUCTURE)
2	AK2	SUPERSTRUCTURE
3	AK3	LIFT SYSTEM (NOT APPLICABLE)
4	AK4	INSTALLED DIESEL POWER
5	AK5	INSTALLED GAS TURBINE POWER
6	AK6	ELECTRICAL SYSTEM
7	AK7	AUXILIARY SYSTEM
8	AK8	OUTFIT AND FURNISHING
9	AK9	SHIPYARD CONSTRUCTION
10	AK10	DESIGN AND ENGINEERING, LEAD SHIP
11	AK11	DESIGN AND ENGINEERING, CLASS SHIP
12	AK12	CONSTRUCTION SERVICES
13	AK13	SHIPYARD PROFIT, LEAD SHIP
14	AK14	SHIPYARD PROFIT, CLASS SHIP
15	AK15	MISCELLANEOUS SURCHARGES, LEAD SHIP
16	AK16	MISCELLANEOUS SURCHANRGES, CLASS SHIP

P48830.PDF [Page: 27 of 80]

APPENDIX B

CREATING A NEW FILE (SAMPLE TERMINAL SESSION)

Following is an example of creation of a new data file using FIVPRE. The user's typed values are underlined for clarity. Commands used are NEW, DUMP, and EXIT.

Note: The command used to run the program FIVPRE depends on which computer and operating system is used. In this example, the command is "RUN FIVPRE".

```
$ RUN FIVPRE

                    FIVPRE

              PRE-PROCESSOR FOR SHOP5

** READING LIBRARY FILE, PLEASE WAIT **

ARE YOU FAMILIAR WITH THIS PROGRAM ?
ENTER 'Y' OR 'N' = > N

        FIVPRE is a program which helps you create and edit the input
        files for SHOP5, the concept exploration model for frigates and
        destroyers.  Using FIVPRE, you can easily define the values of
        the various parameters that SHOP5 requires, change the values of
        these parameters, and save the values in a file, which SHOP5 uses
        as input.

        FIVPRE is a command-oriented program: this means that you type
        commands to tell FIVPRE what to do.  When you are running FIVPRE,
        you will see a prompt 'Command = > ' which means that FIVPRE is
        waiting for you to type in a command.

        To see a list of all the commands you can use in FIVPRE, type
        'HELP' at the 'Command = > ' prompt.  The 'HELP' command can
        also give a more detailed description of a command.

** PUSH 'RETURN' KEY FOR NEXT SCREEN **

        Here is a quick overview of a few basic FIVPRE commands:

        The 'NEW' command starts a chain of questions which will ask you
        for all the parameters you need to create a new SHOP5 input file.
```

The 'DUMP' command saves the current value of all the variables on a file called FIVIN, which is the input file for SHOP5. If you specify a file name in the 'DUMP' command, the data will be saved in that file as well as in FIVIN.

The 'ENTER' command reads an existing file so you can change it.

The 'CHANGE' command modifies existing files which have been ENTERed.

The 'DISPLAY' command displays the value of a variable.

The 'EXIT' command stops the FIVPRE program.

** PUSH 'RETURN' KEY FOR NEXT SCREEN **

If you are not familiar with the SHOP5 variables, type 'NEW' at the 'Command = > ' prompt. After you finish with the 'NEW' command (i.e. when you see the 'Command = > ' prompt again) type 'DUMP' to save the data you have just defined. To end the FIVPRE program, type 'EXIT'.

You may also want to find out about the special input characters. Type 'HELP SPECIAL' at the 'Command = > ' prompt for an explanation.

Command = > NEW

ENTER TITLE = > SAMPLE INPUT 1, 504 SHIPS, USING POST-PROCESSOR

** PROGRAM CONTROL INTEGERS **

MODE

0 = SEARCH
1 = DESCRIBE

MODE = > 0

UNITS

0 = INPUT FPS, OUTPUT FPS
1 = INPUT METRIC, OUTPUT FPS
2 = INPUT FPS, OUTPUT METRIC
3 = INPUT METRIC, OUTPUT METRIC

UNITS = > 0

POST-PROCESSOR

0 = NO, POST-PROCESSOR WILL NOT BE USED
1 = YES, POST-PROCESSOR WILL BE USED

POST-PROCESSOR = > 1

LEGEND

0 = NO LEGEND
1 = LEGEND OF ABBREVIATIONS WRITTEN ON LINEPRINTER

LEGEND = > 0

CHARACTERISTIC DIMENSION

0 - DISPLACEMENT
1 - LENGTH
2 - BEAM
3 - DRAFT

ENTER NUMBER = > 0

DISPLACEMENT:
LOW 3800
HIGH 4200
NUMBER OF VALUES 9

LENGTH/DISPLACEMENT RATIO:
LOW 7.8
HIGH 8.1
NUMBER OF VALUES 4

PRISMATIC COEFFICIENT:
LOW .61
HIGH .62
NUMBER OF VALUES 2

BLOCK COEFFICIENT:
LOW .46
HIGH .52
NUMBER OF VALUES 7

BEAM/DRAFT RATIO:
LOW 3.25
HIGH 3.25
NUMBER OF VALUES 1

```
RANGE AT ENDURANCE SPEED = > 4000
COMBAT SYSTEM WEIGHT = > 350
DESIGN SPEED = > 30
CRUISE SPEED = > 18
ENDURANCE SPEED = > 15
SEAKEEPING SPEED = > 30
SEAKEEPING WAVE HEIGHT = > 9.843

** METHOD CONTROL INTEGERS **

1 - RESIDUARY RESISTANCE          5 - TYPE OF ENGINES
2 - APPENDAGE RESISTANCE          6 - TYPE OF GENERATORS
3 - PROPULSIVE EFFICIENCY         7 - COST FACTORS
4 - STRUCTURAL MATERIAL          99 - ** EXIT **

ENTER NUMBER = > 5


TYPE OF ENGINES

0 - SHOP5 RUBBER ENGINES
1 - USER SUPPLIED RUBBER ENGINES
2 - SHOP5 REAL ENGINES
3 - USER SUPPLIED REAL ENGINES

ENTER NUMBER = > 2


ENGINE DATA BASE

0 - 1980 ENGINE DATA BASE
1 - 1995 ENGINE DATA BASE

ENTER NUMBER = > 0


ENGINE CONFIGURATION

1 - SINGLE GAS TURBINE
2 - COGAG (COmbined Gas And Gas)
3 - COGOG (COmbined Gas Or Gas)
4 - CODAG (COmbined Diesel And Gas)
5 - CODOG (COmbined Diesel Or Gas)
6 - CODAD (COmbined Diesel And Diesel)

ENTER NUMBER = > 2
```

```
CONFIGURATION CHANGE

0 - SHOP5 MUST USE THIS CONFIGURATION
1 - SHOP5 MAY CHANGE CONFIGURATION

ENTER NUMBER = > 0

** METHOD CONTROL INTEGERS **

 1 - RESIDUARY RESISTANCE          5 - TYPE OF ENGINES
 2 - APPENDAGE RESISTANCE          6 - TYPE OF GENERATORS
 3 - PROPULSIVE EFFICIENCY         7 - COST FACTORS
 4 - STRUCTURAL MATERIAL          99 - ** EXIT **

ENTER NUMBER = > 99


OPTIONAL INPUT

 1 - GEOMETRY & COMPLEMENT
 2 - SEAKEEPING
 3 - POWERING AND MACHINERY
 4 - STRUCTURE
 5 - WEIGHT AND STABILITY
 6 - VOLUME
 7 - MISSION
99 - EXIT

ENTER NUMBER = > 1


GEOMETRY AND COMPLEMENT

 1 - MIDSHIPS HULL DEPTH
 2 - WATERPLANE COEFFICIENT
 3 - LONGITUDINAL CENTRE OF BUOYANCY
 4 - LONGITUDINAL CENTRE OF FLOTATION
 5 - COMPLEMENT
 6 - NUMBER OF INTERNAL DECKS
 7 - NUMBER OF WATERTIGHT COMPARTMENTS
 8 - COMPARTMENT STANDARD OF FLOODING
 9 - SUPERSTRUCTURE LENGTH
99 - EXIT

ENTER NUMBER = > 5
     CURRENT VALUE (DEFAULT) = N/(DISP**(2/3)) = 1.0
     USE NEGATIVE NUMBER FOR RATIO;
         POSITIVE NUMBERS FOR FIXED VALUE
ENTER N: COMPLEMENT = > -.9
```

GEOMETRY AND COMPLEMENT

```
 1 - MIDSHIPS HULL DEPTH
 2 - WATERPLANE COEFFICIENT
 3 - LONGITUDINAL CENTRE OF BUOYANCY
 4 - LONGITUDINAL CENTRE OF FLOTATION
 5 - COMPLEMENT
 6 - NUMBER OF INTERNAL DECKS
 7 - NUMBER OF WATERTIGHT COMPARTMENTS
 8 - COMPARTMENT STANDARD OF FLOODING
 9 - SUPERSTRUCTURE LENGTH
99 - EXIT

ENTER NUMBER = > 9
     CURRENT VALUE (DEFAULT) = Ls/L = 0.5
     USE NEGATIVE NUMBER FOR RATIO;
         POSITIVE NUMBERS FOR FIXED VALUE
ENTER Ls:  SUPERSTRUCTURE LENGTH = > -.58
```

GEOMETRY AND COMPLEMENT

```
 1 - MIDSHIPS HULL DEPTH
 2 - WATERPLANE COEFFICIENT
 3 - LONGITUDINAL CENTRE OF BUOYANCY
 4 - LONGITUDINAL CENTRE OF FLOTATION
 5 - COMPLEMENT
 6 - NUMBER OF INTERNAL DECKS
 7 - NUMBER OF WATERTIGHT COMPARTMENTS
 8 - COMPARTMENT STANDARD OF FLOODING
 9 - SUPERSTRUCTURE LENGTH
99 - EXIT

ENTER NUMBER = > 99
```

OPTIONAL INPUT

```
 1 - GEOMETRY & COMPLEMENT
 2 - SEAKEEPING
 3 - POWERING AND MACHINERY
 4 - STRUCTURE
 5 - WEIGHT AND STABILITY
 6 - VOLUME
 7 - MISSION
99 - EXIT

ENTER NUMBER = > 5
```

```
WEIGHT AND STABILITY

 1 - MARGIN: HULL STRUCTURE WEIGHT
 2 - MARGIN: PROPULSION MACHINERY WEIGHT
 3 - MARGIN: ENGINE RAFTING
 4 - MARGIN: GEAR BOX
 5 - MARGIN: ELECTRICAL WEIGHT
 6 - MARGIN: AUXILIARIES WEIGHT
 7 - MARGIN: OUTFIT & FURNISHING WEIGHT
 8 - MARGIN: BASIC WEIGHT
 9 - MARGIN: DISPOSABLE WEIGHT
10 - VCG OF EXTRA BASIC WEIGHT
11 - VCG OF COMBAT WEIGHT
99 - EXIT

ENTER NUMBER = > 3
     CURRENT VALUE (DEFAULT) = - 0.0
     USE NEGATIVE NUMBER FOR RATIO;
         POSITIVE NUMBERS FOR FIXED VALUE
ENTER RAFT: MARGIN: MAIN ENGINE RAFTING = > 50


WEIGHT AND STABILITY

 1 - MARGIN: HULL STRUCTURE WEIGHT
 2 - MARGIN: PROPULSION MACHINERY WEIGHT
 3 - MARGIN: ENGINE RAFTING
 4 - MARGIN: GEAR BOX
 5 - MARGIN: ELECTRICAL WEIGHT
 6 - MARGIN: AUXILIARIES WEIGHT
 7 - MARGIN: OUTFIT & FURNISHING WEIGHT
 8 - MARGIN: BASIC WEIGHT
 9 - MARGIN: DISPOSABLE WEIGHT
10 - VCG OF EXTRA BASIC WEIGHT
11 - VCG OF COMBAT WEIGHT
99 - EXIT

ENTER NUMBER = > 8
     CURRENT VALUE (DEFAULT) = - 0.0
     USE NEGATIVE NUMBER FOR RATIO;
         POSITIVE NUMBERS FOR FIXED VALUE
ENTER dWb: MARGIN: BASIC WEIGHT (WB=W1+W2+W3+W5+W6) = > -.025
```

```
WEIGHT AND STABILITY

 1 - MARGIN: HULL STRUCTURE WEIGHT
 2 - MARGIN: PROPULSION MACHINERY WEIGHT
 3 - MARGIN: ENGINE RAFTING
 4 - MARGIN: GEAR BOX
 5 - MARGIN: ELECTRICAL WEIGHT
 6 - MARGIN: AUXILIARIES WEIGHT
 7 - MARGIN: OUTFIT & FURNISHING WEIGHT
 8 - MARGIN: BASIC WEIGHT
 9 - MARGIN: DISPOSABLE WEIGHT
10 - VCG OF EXTRA BASIC WEIGHT
11 - VCG OF COMBAT WEIGHT
99 - EXIT

ENTER NUMBER = > 99


OPTIONAL INPUT

 1 - GEOMETRY & COMPLEMENT
 2 - SEAKEEPING
 3 - POWERING AND MACHINERY
 4 - STRUCTURE
 5 - WEIGHT AND STABILITY
 6 - VOLUME
 7 - MISSION
99 - EXIT

ENTER NUMBER = > 6


VOLUME

 1 - SUPERSTRUCTURE VOLUME
 2 - MARGIN: MACHINERY VOLUME
 3 - MARGIN: SHIP SYSTEMS & OUTFIT VOLUME
 4 - MARGIN: BASIC VOLUME
 5 - MARGIN: PERSONNEL VOLUME
99 - EXIT

ENTER NUMBER = > 1
      CURRENT VALUE (DEFAULT) = Vs/Vt = 0.25
      USE NEGATIVE NUMBER FOR RATIO;
          POSITIVE NUMBERS FOR FIXED VALUE
ENTER Vs: SUPERSTRUCTURE VOLUME = > -.28
```

```
VOLUME

 1 - SUPERSTRUCTURE VOLUME
 2 - MARGIN: MACHINERY VOLUME
 3 - MARGIN: SHIP SYSTEMS & OUTFIT VOLUME
 4 - MARGIN: BASIC VOLUME
 5 - MARGIN: PERSONNEL VOLUME
99 - EXIT

ENTER NUMBER = > 99


OPTIONAL INPUT

 1 - GEOMETRY & COMPLEMENT
 2 - SEAKEEPING
 3 - POWERING AND MACHINERY
 4 - STRUCTURE
 5 - WEIGHT AND STABILITY
 6 - VOLUME
 7 - MISSION
99 - EXIT

 ENTER NUMBER = > 99


REJECTION CRITERIA

 1 - NUMBER OF DECK WETNESS PER HOUR
 2 - VERTICAL ACCELERATION AT STATION 4
 3 - SLAM FORCE/DISPLACEMENT
 4 - SEAKEEPING EFFECTIVENESS
 5 - OPC AT ENDURANCE SPEED
 6 - VERTICAL CENTRE OF GRAVITY
 7 - COMBAT SYSTEM VOLUME
99 - EXIT

ENTER NUMBER = > 7
       CURRENT VALUE (DEFAULT) = 0.125
VOLUME: MIN. Vc (USE NEGATIVE NUMBER FOR Vc/Vt) = > -.15
```

```
REJECTION CRITERIA

 1 - NUMBER OF DECK WETNESS PER HOUR
 2 - VERTICAL ACCELERATION AT STATION 4
 3 - SLAM FORCE/DISPLACEMENT
 4 - SEAKEEPING EFFECTIVENESS
 5 - OPC AT ENDURANCE SPEED
 6 - VERTICAL CENTRE OF GRAVITY
 7 - COMBAT SYSTEM VOLUME
99 - EXIT

ENTER NUMBER = > 99

*** END OF SEQUENTIAL INPUT ***
Command = > DUMP
FILE NAME = > SAMPLE
     WRITING DATA TO SAMPLE

     WRITING DATA TO FIVIN

Command = > EXIT

                              **** FINISHED ****

FORTRAN STOP
```

APPENDIX C

EDITING AN EXISTING FILE (SAMPLE TERMINAL SESSION)

Following is an example of editing an existing file, EX2, to create a file named EX3. The user's typed values are underlined for clarity. Commands used are HELP, FIND, ENTER, DISPLAY, CHANGE, DUMP, and EXIT.

Note: The command used to run the program FIVPRE depends on which computer and operating system is used. In this example, the command is "RUN FIVPRE".

```
$ RUN FIVPRE

                                FIVPRE

                        PRE-PROCESSOR FOR SHOP5


** READING LIBRARY FILE, PLEASE WAIT **

ARE YOU FAMILIAR WITH THIS PROGRAM ?
ENTER 'Y' OR 'N' = > Y
Command = > HELP

*** FIVPRE COMMANDS AND TOPICS ***

        CHANGE                  EXIT
        DEFAULT                 FIND
        DISPLAY                 HELP
        DUMP                    NEW
        ENTER                   Special

For help on a particular command, type
        HELP command

Command = > HELP FIND

        FIND

        This command is used to find the name of a data file, based on
        its title record.  After entering the command FIND, the user is
        prompted for a search key.  If the user hits the return key
```

without entering a search string, the program will print all the entries in the library file; otherwise, the program prints the title and file name of the files in its library which contain the search string in their title.

Format:

FIND

```
Command = > FIND
ENTER SEARCH STRING
        (TO DISPLAY ALL ENTRIES, JUST HIT RETURN) = > HELI
SHOP5 EXAMPLE 1, 4000 TON SHIP, HELICOPTER
        DATA FILE = EX1
SHOP5 EXAMPLE 2, 4000 TON SHIP, HELICOPTER, COGAG
        DATA FILE = EX2
Command = > ENTER EX2

READING DATA FROM EX2

Command = > DISPLAY METHOD CONTROL INTEGERS


METHOD CONTROL INTEGERS

CHARACTERISTIC DIMENSION = Displacement
RESIDUARY RESISTANCE = NRC FSS Data
APPENDAGE RESISTANCE = SHOP5 Calculations
PROPULSIVE COEFFICIENT = SHOP5 Calculations, C.P. Propeller
STRUCTURAL MATERIAL = Homogenous Hull and Superstructure (Steel)
*PROPULSION SYSTEM = SHOP5 Engine Data Base
ENGINE CONFIGURATION SELECTION = May Change Selected Engine Configuration
GENERATORS = SHOP5 Gas Turbine Generators
COST FACTORS = SHOP5 Cost Factors (1977 Dollars)
Command = > DISP ENGINE DATA
** AMBIGUOUS INPUT **

1 - IFUTUR: CHOICE OF SHOP5 ENGINE DATA BASE
2 - RECORD 9: ENGINE ALGORITHM DATA
ENTER SELECTION NUMBER (OR 99 FOR EXIT) = > 2


PROPULSION SYSTEM

Engine Configuration = COGAG
1980 Engine Data Base
Hierarchy is NOISE
Command = > CHANGE ENGINE CONFIG
```

```
** AMBIGUOUS INPUT **
1  ICHOOS: ENGINE CONFIGURATION SELECTION
2  ITYPE: ENGINE CONFIGURATION PER SHAFT
ENTER SELECTION NUMBER (OR 99 FOR EXIT) => 2

ENGINE CONFIGURATION

1 - SINGLE GAS TURBINE
2 - COGAG (COmbined Gas And Gas)
3 - COGOG (COmbined Gas Or Gas)
4 - CODAG (COmbined Diesel And Gas)
5 - CODOG (COmbined Diesel Or Gas)
6 - CODAD (COmbined Diesel And Diesel)

ENTER NUMBER => 4

ENGINE CONFIGURATION SELECTION

0 - SHOP5 MUST USE THIS CONFIGURATION
1 - SHOP5 MAY CHANGE CONFIGURATION

ENTER NUMBER => 1

GEARBOX HIERARCHY

0 - SHOP5 GEARBOX HIERARCHY
1 - USER SUPPLIED GEARBOX HIERARCHY

ENTER NUMBER => 0

GEARBOX HIERARCHIES

        GEARBOX SELECTION BEGINS AT THE BOTTOM OF A COLUMN AND BUBBLES
        UPWARDS, UNTIL THE USER-SELECTED CONFIGURATION IS ENCOUNTERED.
        SHOP5 ATTEMPTS TO SATISFY POWER REQUIREMENTS WITH THIS
        CONFIGURATION;  IF THIS CANNOT BE ACCOMPLISHED, THE NEXT
        (UPWARDS) CONFIGURATION IS TRIED, UNTIL EITHER SATISFACTORY
        ENGINES ARE FOUND, OR ALL POSSIBILITIES ARE TRIED.
```

```
SELECT ONE OF THESE HIERARCHIES

                NOISE = 0      GEARING = 1      POWER = 2

                               CODAG            COGAG
                CODAG          COGAG            CODAG
                COGAG          COGOG            COGOG
                CODOG          CODOG            CODOG
                COGOG          CODAD            CODAD
                SINGLE         SINGLE           SINGLE

ENTER NUMBER = > 0
Command = > DISPLAY METHOD CONTROL INTEGERS


METHOD CONTROL INTEGERS

CHARACTERISTIC DIMENSION = Displacement
RESIDUARY RESISTANCE = NRC FSS Data
APPENDAGE RESISTANCE = SHOP5 Calculations
PROPULSIVE COEFFICIENT = SHOP5 Calculations, C.P. Propeller
STRUCTURAL MATERIAL = Homogenous Hull and Superstructure (Steel)
*PROPULSION SYSTEM = SHOP5 Engine Data Base
ENGINE CONFIGURATION SELECTION = May Change Selected Engine Configuration
GENERATORS = SHOP5 Gas Turbine Generators
COST FACTORS = SHOP5 Cost Factors (1977 Dollars)
Command = > CHANGE GENERATOR TYPE


TYPE OF GENERATORS

0 - SHOP5 GAS TURBINE GENERATORS
1 - SHOP5 DIESEL GENERATORS
2 - USER SUPPLIED GENERATORS

ENTER NUMBER = > 1

Command = > DISPLAY OPTIONAL INPUT


RE-DEFINED OPTIONAL VARIABLES ARE:

N: COMPLEMENT = 200.0000
Ls: SUPERSTRUCTURE LENGTH = -0.5750
RAFT: MARGIN: MAIN ENGINE RAFTING = 50.0000
dWb: MARGIN: BASIC WEIGHT (WB=W1+W2+W3+W5+W6) = -0.0200
Vs: SUPERSTRUCTURE VOLUME = -0.2900
```

```
Command = > CHANGE LS
DEFAULT VALUE IS Ls/L = 0.5
CURRENT VALUE IS -0.5750
         USE NEGATIVE NUMBER FOR RATIO;
             POSITIVE NUMBERS FOR FIXED VALUE
ENTER Ls: SUPERSTRUCTURE LENGTH = > DEF
Command = > CHANGE VS
DEFAULT VALUE IS Vs/Vt = 0.25
CURRENT VALUE IS -0.2900
         USE NEGATIVE NUMBER FOR RATIO;
             POSITIVE NUMBERS FOR FIXED VALUE
ENTER Vs: SUPERSTRUCTURE VOLUME = > DEF
Command = > CHANGE GEAR
CURRENT VALUE (DEFAULT) =  0.0
         USE NEGATIVE NUMBER FOR RATIO;
             POSITIVE NUMBERS FOR FIXED VALUE
ENTER GEAR: MARGIN: GEAR BOX WEIGHT = > 50
Command = > CHANGE KGC
CURRENT VALUE (DEFAULT) = KGc/D = 0.70
         USE NEGATIVE NUMBER FOR RATIO;
             POSITIVE NUMBERS FOR FIXED VALUE
ENTER KGc: KG OF Wc = > -.70
Command = > DISPLAY OPT INPUT


RE-DEFINED OPTIONAL VARIABLES ARE:

N: COMPLEMENT = 200.0000
RAFT: MARGIN: MAIN ENGINE RAFTING = 50.0000
GEAR: MARGIN: GEAR BOX WEIGHT = 50.0000
dWb: MARGIN: BASIC WEIGHT (WB=W1+W2+W3+W5+W6) = -0.0200
KGc: KG OF Wc = -0.7000
Command = > DISPLAY ALL


TITLE = SHOP5 EXAMPLE 2, 4000 TON SHIP, HELICOPTER, COGAG


PROGRAM CONTROL INTEGERS

MODE = Describe
INPUT/OUTPUT UNITS = Input - FPS, Output - FPS
DISK FILES FOR POST-PROCESSOR ? = No
LINEPRINTER OUTPUT ? = No
LEGEND OF ABBREVIATIONS WITH OUTPUT ? = No

** PUSH 'RETURN' KEY FOR NEXT SCREEN **
```

PRIMARY INPUT

	SHIP NUMBER
	1
DISPL	4000.0
CRM	8.10
CP	0.620
CB	0.500
B/T	3.200
WC	400.00
VD	30.00
VC	18.00
VE	15.00
VW	30.00
HW	9.840

** PUSH 'RETURN' KEY FOR NEXT SCREEN **

METHOD CONTROL INTEGERS

CHARACTERISTIC DIMENSION = Displacement
RESIDUARY RESISTANCE = NRC FSS Data
APPENDAGE RESISTANCE = SHOP5 Calculations
PROPULSIVE COEFFICIENT = SHOP5 Calculations, C.P. Propeller
STRUCTURAL MATERIAL = Homogenous Hull and Superstructure (Steel)
*PROPULSION SYSTEM = SHOP5 Engine Data Base
ENGINE CONFIGURATION SELECTION = May Change Selected Engine Configuration
*GENERATORS = SHOP5 Diesel Generators
COST FACTORS = SHOP5 Cost Factors (1977 Dollars)

** PUSH 'RETURN' KEY FOR NEXT SCREEN **

RE-DEFINED OPTIONAL VARIABLES ARE:

N: COMPLEMENT = 200.0000
RAFT: MARGIN: MAIN ENGINE RAFTING = 50.0000
GEAR: MARGIN: GEAR BOX WEIGHT = 50.0000
dWb: MARGIN: BASIC WEIGHT (WB=W1+W2+W3+W5+W6) = -0.0200
KGc: KG OF Wc = -0.7000

```
NO RE-DEFINED SEAKEEPING CRITERIA

** PUSH 'RETURN' KEY FOR NEXT SCREEN **


PROPULSION SYSTEM

Engine Configuration = CODAG
1980 Engine Data Base
Hierarchy is NOISE
Command = > CHANGE TITLE
ENTER TITLE = > SHOP5 EXAMPLE 3, 4000 TON SHIP, CODAG
Command = > DISP TITLE


TITLE = SHOP5 EXAMPLE 3, 4000 TON SHIP, CODAG

Command = > DUMP
FILE NAME = > EX3

** THIS FILE ALREADY EXISTS **

DO YOU WANT TO REPLACE IT WITH THE CURRENT FILE ?
IF NOT, TRY THE 'DUMP' COMMAND AGAIN, WITH A NEW FILE NAME

REPLACE ? ENTER 'Y' or 'N' = > Y
WRITING DATA TO EX3

WRITING DATA TO FIVIN

Command = > EXIT

                              **** FINISHED ****

FORTRAN STOP
```

APPENDIX D

MACHINE DEPENDENCIES

FIVPRE is written in VAX-11 FORTRAN V3.0 on a VAX-11/750 computer, using the VMS operating system. It conforms to the ANSI standard FORTRAN of 1977, except for the following features:

FIVPRE uses the character $ in FORMAT statements to suppress a line feed after a WRITE statement. This feature is used to position the cursor on the same line as the prompt after a prompt is written on the terminal. Removing the $ from the FORMAT statements will not affect the execution of the program, except that responses to the prompts will have to be entered on a new line.

The OPEN and CLOSE statements are Standard FORTRAN 77 in syntax, but other computers may not implement these statements in the same way that the VAX does. The parameter IOSTAT = IOVAL is used in the OPEN statement to detect an error while trying to open a file; FIVPRE assumes that there has been no error if IOVAL has the value of 0 after the OPEN statement has executed.

The device numbers used in READ and WRITE statements are NTTY, NDISK, and NDISK2 (see Appendix E.1.), and their values are set in the main program unit FIVPRE. The device numbers for terminal and disk on other machines may be different from those on the VAX.

APPENDIX E

GENERAL PROGRAM STRUCTURE OF FIVPRE

E.1 DATA STRUCTURES

FIVPRE uses common blocks to hold the values of all the variables for SHOP5 input. Common blocks are also used to hold flags for these input variables to indicate whether or not a variable has been defined.

The common blocks for SHOP5 input variables are:

```
COMMON/NO1/TITLE
COMMON/NO2/MODE,INOUT,IPOST,ILPT,ILEGND
COMMON/NO3A1/XXXLO,XXXHI,I1,CRMLO,CRMHI,I2,CPLO,CPHI,I3,CBLO,
        CBHI,I4,BOTLO,BOTHI,I5,E,WC,VD,VC,VE,VW,HW
COMMON/NO3A2/XXX(11),CRM(11),CP(11),CB(11),BOT(11),WC2(11),
        VD2(11),VC2(11),VE2(11),VW2(11),HW2(11),NSHIPS
COMMON/NO4/IDIMEN,IRESID,IAPPND,IPROP,ISTRUC,IENGIN,ICHOOS,IGEN, ICOST
COMMON/NO5/OPTVAR(44)
COMMON/NO6/REJECV(11)
COMMON/NO7/CAPPU
COMMON/NO8/ETAD,ETAC,ETAE
COMMON/NO9/SPW2,SFCR,ITYPE,IFUTUR,IHIAR,HIARK(6),SPW2B,NTURB,
        POWT(20),SFCT(20),WTT(20),NDIES,POWD(20),SFCD(20),WTD(20)
COMMON/NO9CH/NAMTUR(20),NAMDIE(20)
COMMON/NO10/SFCG,SPGEN
COMMON/NO11/AK(16),NCOST
COMMON/NUMBER/NCON,NOPTN,NLIM
```

The common blocks for the flags are:

```
COMMON/NO1A/FLAG1
COMMON/NO2A/FLAG2(5)
COMMON/NO3A/FLAG3S(12),FLAG3D(11,11)
COMMON/NO4A/FLAG4(9)
COMMON/NO5A/FLAG5(44)
COMMON/NO6A/FLAG6(11)
COMMON/NO7A/FLAG7
COMMON/NO8A/FLAG8
COMMON/NO9A/FLAG9(6)
COMMON/NO10A/FLAG10
COMMON/NO11A/FLAG11(16)
```

The other important common blocks are those which set the device numbers for input and output:

```
COMMON/DEVICE/NTTY,NDISK,NDISK2
```

Other common blocks are described with the subroutines which use them.

E.2 ALGORITHM OF MAIN PROGRAM UNIT FIVRE

In general, the body of the main program unit is a loop which accepts a command from the terminal, performs the appropriate action, accepts another command from the terminal, performs the appropriate action, and so on. The loop ends when the command is "EXIT".

The pseudocode description gives more detail:

```
Begin
$ Initialize all flags and variables
$ Read in library file and update library data
  Repeat
     Accept a line of input from the terminal
   $ Process the line of input (break it up into a word list)
   $ Determine which command is meant
     If command is "CHANGE" then
      $ Change the value of the specified variable
     Else if command is "DEFAULT" then
      $ Display the default value of the specified variable
     Else if command is "DISPLAY" then
      $ Display the current value of the specified variable
     Else if command is "DUMP" then
      $ Write the data to a disk file
     Else if command is "ENTER" then
      $ Read in data from a disk file
     Else if command is "FIND" then
      $ Look for search key in the list of titles
     Else if command is "HELP" then
      $ List all the commands if no command is specified, or
        list the information on the specified command
     Else if command is "NEW" then
      $ Change the values of all the variables
     End if
  Until command is "EXIT"
  Write the library data to the library file
End
```

The lines marked with a dollar sign ($) indicate that a subroutine is called to perform that action.

The subroutines do most of the work in the program. They fall into 4 main categories:

(1) Command processing subroutines.
There is only 1 subroutine in this category. This subroutine takes a word of user input and determines which command it corresponds to.

(2) Dictionary subroutines.
These subroutines access the dictionary, a data block which contains the names and descriptions of all the SHOP5 input variables. One of the subroutines determines which variable is being specified by the user; another subroutine finds the description of a variable, given its code; another finds the description of the default value of a variable, given its code.

(3) Command execution subroutines.
These subroutines carry out the action specified by a command, e.g. change the value of variable, dump the data to a file.

(4) Utility subroutines.
These subroutines perform a variety of tasks. Some of these tasks are: breaking a line of input into a list of words, converting a character string to a numeric value, capitalizing the alphabetic characters in a string, stripping leading blanks from a character string.

All these subroutines are described in more detail in following appendices.

APPENDIX F

COMMAND PROCESSING SUBROUTINES

This category of subroutines contains only one subroutine, the integer function ICMDS.

Integer Function ICMDS

Purpose:

ICMDS takes a word (a character string) and looks through a dictionary to find a command that matches the word. The dictionary is an array containing the command words. A word is considered to match a command if the letters of the word are the same as the first letters of the command. For example, "EX" is a match for "EXIT". If there is no match, ICMDS is given the value -1 and control is returned to the main program. If there is more than one match, ICMDS prints a message and returns to the main program with a value of -2. If there is a unique match, the value of ICMDS is the position of the command in the array.

Arguments:

WORD, LENGTH

where

WORD = the word to be tested
LENGTH = the length of the word

Algorithm:

ICMDS looks for WORD in each entry of the command dictionary, using the FORTRAN function INDEX, and counts the number of entries which start with the character string in WORD. If the count is 0, the value of ICMDS is set to -1. If the count is greater than 1, ICMDS prints a message saying that the command is not unique, and then prints a list of the all the commands which could be specified by WORD. If the count is 1, the value of ICMDS is set to the position of the matching command in the array.

APPENDIX G

THE FIVPRE DICTIONARIES AND ASSOCIATED SUBROUTINES

G.1 DICTIONARY STRUCTURE

The FIVPRE dictionaries are character arrays. There are three dictionaries: the abbreviation dictionary, the description dictionary, and the default value dictionary. Each dictionary entry refers to a SHOP5 input variable through a 10-character prefix. The structure of the prefix is

RR.SS.VVV.

where RR is the record number of the variable, SS is the subrecord number of the variable, and VVV is the variable code of the variable.

For example, a typical dictionary entry would be

'05.02.000.Cw: WATERPLANE COEFFICIENT'.

For this entry, the record number is 5 and the subrecord number is 2.

There are several reasons for using the prefix to identify a variable. One reason is that the dictionary does not need to be ordered when the prefix is used. The identification of the variable is independent of its position in the array. Therefore adding new entries to the dictionary is easy, since the dictionary does not have to be rearranged each time a new entry is added. Another reason for using the prefix is that more than one entry can refer to the same variable. This feature is especially useful in the dictionary of abbreviations, since there may be several abbreviations which refer to the same variable.

The FIVPRE dictionaries are contained in the common blocks

```
COMMON/NAMLST/SNAMES,LNAMES
COMMON/VALUES/DEFVAL
```

where SNAMES is the abbreviation dictionary, LNAMES is the description dictionary, and DEFVAL is the default value dictionary.

G.2 DICTIONARY SUBROUTINES

G.2.1 Subroutine DICT

Purpose:

DICT uses the abbreviation dictionary and the description dictionary to identify which variable is being specified by a word list. DICT consults the abbreviation dictionary first, looking for an exact match. The method used is a simple linear search. If no match is found, DICT then searches through the description dictionary. A series of linear searches is used to look through the description dictionary. If a match is found, DICT returns the codes for record number, subrecord number, and variable number. If there is more than one match, but less than 15 matches, DICT displays the possible choices and asks the user to pick one. If there are more than 15 matches, DICT prints a message telling the user that the description is too ambiguous. If there are no matches, DICT sets the flag FOUND to false.

Arguments:

BEGIN, RECNUM, SUBREC, VARCOD, FOUND

where

BEGIN = position in word list to start processing from
RECNUM = character code for record number
SUBREC = character code for subrecord number
VARCOD = character code for variable number
FOUND = a flag which returns true if a match is found, false if no match is found

Common blocks:

COMMON/NAMLST/SNAMES,LNAMES	(see Appendix G.1.)
COMMON/CHARS/WRDLST	(see Appendix I.5.)
COMMON/WRDVAL/WRDLEN,NOWRDS	(see Appendix I.5.)

Algorithm:

```
Begin
   Set search string = first word to be processed in WRDLST
   Repeat
      Search through abbreviation dictionary
      If no match, add next word in word list to search string
   Until a match is found or search string is too long
```

```
      If match is found, then
         Set values of RECNUM, SUBREC, and VARCOD
         Set FOUND to true
         RETURN

      Else
         Set current word = first word to be processed in WRDLST
         While not at end of WRDLST and more than 1 matching entry do
            Search through description dictionary and set dictionary flag
            for each entry which doesn't contain the current word
            Set current word = next word in WRDLST
         End while
         If number of matching entries = 1 then
            Set values of RECNUM, SUBREC, and VARCOD
            Set FOUND to true
            RETURN
         End if
         If number of matching entries > 15 then
            Write a message
            Set FOUND to false
            RETURN
         End if
         If number of matching entries > 1 and < 15 then
            Remove duplicates from matching entries (i.e. choices which
            refer to the same variable)
            Write list of choices and read user's choice
            If choice is "none of the above" then
               Set FOUND to false
               RETURN
            Else
               Set values of RECNUM, SUBREC, and VARCOD
               Set FOUND to true
               RETURN
            End if
         End if
         If number of matching entries = 0 then
            Write message
            RETURN
         End if
      End if
End
```

G.2.2 Subroutine GETDES

Purpose:

GETDES searches the description dictionary to find the description corresponding to the given record number, subrecord number, and variable number.

Arguments:

RECNUM, SUBREC, VARCOD, LENGTH, DESCR

where

RECNUM = character code for record number
SUBREC = character code for subrecord number
VARCOD = character code for variable number
LENGTH = number of characters in the description (DESCR)
DESCR = the character string which is the description of the variable

Common blocks:

COMMON/NAMLST/SNAMES,LNAMES (see Appendix G.1.)

Algorithm:

GETDES creates a search string by concatenating RECNUM, a period, SUBREC, another period, VARCOD, and another period. Then GETDES uses a linear search through the entries in the description dictionary LNAMES and returns the first description which contains the search string in its prefix.

G.2.3 Subroutine GETDFT

Purpose:

GETDFT is similar to GETDES. It searches through the default value dictionary to find the entry which contains the given record number, subrecord number, and variable number in its prefix.

Arguments:

RECNUM, SUBREC, VARCOD, LENGTH, DESCR

where

RECNUM = character code for record number
SUBREC = character code for subrecord number
VARCOD = character code for variable number

LENGTH = number of characters in the description of the default value
DESCR = the character string which is the description of the variable's default value

Common blocks:

COMMON/VALUES/DEFVAL (see Appendix G.1.)

Algorithm:

GETDFT first check the values of RECNUM and SUBREC to determine whether the variable is ICHOOS or DIA. The default values of these two variables depends on the value of MODE (for ICHOOS) and NSH (for DIA). If the variable is not ICHOOS or DIA, GETDFT creates a search string by concatenating RECNUM, a period, SUBREC, another period, VARCOD, and another period. Then GETDFT uses a linear search through the entries in the default value dictionary DEFVAL and returns the first description which contains the search string in its prefix.

APPENDIX H

COMMAND EXECUTION SUBROUTINES

These subroutines carry out the actions which are specified by the user's commands.

H.1 SUBROUTINE CHANGE AND ASSOCIATED SUBROUTINES

These subroutines correspond to the CHANGE and NEW commands. They define or re-define the values of the SHOP5 input variables. For each record, there is a subroutine, or set of subroutines, which defines the variables in that record. These subroutines have the names CRECn, where n is the number of the record. For example, CREC4 is the subroutine which defines or changes the values of Record 4, the Method Control Integers. The subroutine CHANGE controls which of the CRECn subroutines is called to change the value of a variable.

The CHANGE and CRECn subroutines can be called in one of two modes: either sequential or non-sequential. In sequential mode, the flow of logic is sequential. In non-sequential mode, the flow of logic jumps to the appropriate section of code and then control is returned to the main program. This is accomplished by having entry points at the beginning of each section of code and a check for sequential mode at the end of each section of code. If the mode is non-sequential, the flow of logic jumps to the end of the subroutine after completing the specified section of code. For example, here is a sample algorithm of a hypothetical program with a similar organization to CHANGE:

```
BEGIN
        If sequential mode go to Section 1
        If non-sequential, find which section should be executed
        If 1, go to Section 1
        If 2, go to Section 2
        If 3, go to Section 3
        If 4, go to Section 4
        Section 1
           .
           .
           .
        If non-sequential mode, go to END
        Section 2
           .
           .
           .
```

```
        If non-sequential mode, go to END
        Section 3
           .
           .
           .
        If non-sequential mode, go to END
        Section 4
           .
           .
           .
        If non-sequential mode, go to END
END
```

This program in sequential mode would execute Section 1, thenSection 2, then Section 3, then Section 4. In non-sequential mode, only one of these sections would be executed: the program would jump to the relevant section, execute the code until it reached the statement testing for non-sequential mode, then jump to the end of the program.

If CHANGE is called in the sequential mode, all the CRECn subroutines are called; this corresponds to the NEW command. In the non-sequential mode, CHANGE determines which record the variable belongs to and jumps to the appropriate section of code, which calls the corresponding CRECn subroutine.

In the sequential mode for a CRECn subroutine, all the variables in a record are changed or defined; this corresponds to a "CHANGE record name" command. The CRECn subroutine in the non-sequential mode determines which subrecord the variable belongs to, jumps to the appropriate section of code, which changes the value of the variable.

When in non-sequential mode, the CHANGE subroutines and the CRECn subroutines use the FORTRAN computed GO TO statement to transfer control to the appropriate sections of code.

The structure of some of the CRECn subroutines may vary from the structure described above. This was necessary because the relationships between some variables require a "ripple" effect: when one of the variables is changed, the other variables related to it must be changed as well.

H.1.1 Subroutine CHANGE

Purpose:

This subroutine coordinates the subroutines which change the values of the SHOP5 input variables.

Arguments:

NEW

where

NEW = an integer which determines whether sequential or non-sequential mode is used
NEW = 1 invokes sequential mode
NEW = 0 invokes non-sequential mode

Common blocks:

```
COMMON/DEVICE/NTTY,NDISK,NDISK2
COMMON/NO2/MODE,INOUT,IPOST,ILPT,ILEGND  (see Appendix E.1.)

COMMON/CHARS/WRDLST
COMMON/WRDVAL/WRDLEN,NOWRDS              (see Appendix I.5.)
COMMON/INFO/BEGIN
```

The value of BEGIN is set in the main program unit FIVPRE. BEGIN is the starting position for processing the word list WRDLST (see Appendix G.2.1. Subroutine DICT).

Algorithm:

```
Begin
    If sequential mode is specified, then
       call INITIA to initialize variables
    Else
       Call DICT to determine which variable to change
       Find which record the variable belongs to
       Jump to the appropriate section of code, depending on the record
       number
    End if
    {Record 1}
    Call CREC1
    If non-sequential mode, RETURN
    {Record 2}
    Call CREC2
    If non-sequential mode, RETURN
    {Record 3}
    Call CREC3
    If non-sequential mode, RETURN
    {Record 4}
    Call CREC4
    If non-sequential mode, RETURN
```

```
    {Record 5}
    Call CREC5
    If non-sequential mode, RETURN
    {Record 6}
    Call CREC6
    If non-sequential mode, RETURN
    RETURN
    {Record 7}
    CALL CREC7
    If non-sequential mode, RETURN
    {Record 8}
    Call CREC8
    If non-sequential mode, RETURN
    {Record 9}
    Call CREC9
    If non-sequential mode, RETURN
    {Record 10}
    Call CREC10
    If non-sequential mode, RETURN
    {Record 11}
    Call CREC11
    If non-sequential mode, RETURN
End
```

H.1.2 The CRECn Subroutines

Most of the CRECn subroutines have the arguments

SUBREC, VARCOD, IALL

where

SUBREC = the character code which identifies the subrecord number (e.g. the character code '02' identifies subrecord 2)
VARCOD = the character code which identifies the variable number
IALL = an integer whose value determines whether sequential or non-sequential mode is used

All the CRECn subroutines have the common block

```
COMMON/DEVICE/NTTY,NDISK,NDISK2
```

(see Appendix E.1.)

In addition, each CRECn subroutine contains the common blocks for the variables which the subroutine changes or defines.

A few subroutines, namely CREC4, C3SUB1 and C3SUB1 (which are subroutines used by CREC3) use the common block

```
COMMON/CHARDM/CHRDIM,NDIM
```

where CHRDIM is a character array (CHRDIM(4)*12) and NDIM is an integer. CHRDIM holds the strings 'DISPLACEMENT', 'LENGTH', 'BEAM', and 'DRAFT'. NDIM is 1 if the characteristic dimension is displacement, 2 if the characteristic dimension is length, 3 if it is beam, and 4 if it is draft.

The program structure for most of the CRECn subroutines is basically the sequential/nonsequential structure described in Appendix H.1. above. Exceptions are noted below.

H.1.2.1 Subroutine CREC3

This subroutine decides which of two subroutines should be called to change variables in Record 3. The two subroutines are called C3SUB1 and C3SUB2. Subroutine C3SUB1 is used when the program control integer MODE is 0 (indicating search mode in SHOP5); subroutine C3SUB2 is used when MODE is 1 (indicating describe mode in SHOP5).

Arguments:

SUBREC, VARCOD, IALL, ISWTCH

where

SUBREC = the character code which identifies the subrecord number (e.g. the character code '02' identifies subrecord 2)
VARCOD = the character code which identifies the variable number
IALL = an integer whose value determines whether sequential or non-sequential mode is used
ISWTCH = an integer whose value determines whether FIVPRE should give the user the option of transferring data from one mode to another

The variable ISWTCH requires more explanation. If the user changes the value of MODE, he must re-define the primary input variables for Record 3. If ISWTCH is 1, the subroutine will ask the user if he wants to keep the same operational requirements that he had defined for the previous MODE. If the answer is yes, the program will transfer the data from the primary input variables for one mode to the corresponding variables for the other mode.

The subroutine CREC3 does not actually use the value of ISWTCH; it merely passes the value on to the subroutines C3SUB1 and C3SUB2.

H.1.2.1.1. Subroutines C3SUB1 and C3SUB2

Subroutine C3SUB1 has the same arguments as CREC3 subroutines (i.e. SUBREC,VARCOD,IALL,ISWTCH). It has the general sequential/non-sequential program structure described in Appendix H.1.

Subroutine C3SUB2 also has the same arguments as CREC3 subroutines, but the program structure varies from the sequential/nonsequential structure described Appendix H.1. C3SUB2 does contain sections of code which are accessed either sequentially or non-sequentially, and the non-sequential access is the same as for the other CRECn subroutines. However, these sections of code are enclosed in one large loop, which is used for sequential access. Each repetition of the loop defines or changes the primary input for one ship. For sequential mode, the loop is repeated until the values for all the ships have been defined.

H.1.2.2 Subroutines CREC5, CREC6, And CREC11

The subroutines CREC5, CREC6, and CREC11 have similar structures, so they will be discussed together. They all have the standard CRECn arguments (i.e.SUBREC,VARCOD,IALL), but they do not have the standard CRECn program structure. The algorithm they use is as follows:

```
Begin
    Write a menu of variables which can be re-defined
    Ask the user which variable he wishes to re-define
    If the user wants to exit the re-definition of variables, then
       RETURN
    Convert the number of the variable to the character code
    Call GETDES to get the description of the variable
    Write a prompt for the new value of the variable, using the
    description from GETDES
    Read the new value from the terminal
    Go to the beginning of the subroutine
End
```

This algorithm was used to avoid too much repetitious code in the subroutines CREC5, CREC6, and CREC11.

H.1.2.3 Subroutine CREC9

Subroutine CREC9 changes or defines the values for the variables in Record 9. The program structure is not the same as that of the other CRECn subroutines, because the subrecords of Record 9 are not organized in a straightforward manner. The variables used for Record 9 depend on the value of IENGIN (a method control variable). If IENGIN has the value 1, then Record 9 is composed of the variables in Subrecord 1. If IENGIN is 2, then Record 9 is composed of the variables in Subrecords 2, 3, and 4. If IENGIN is 3, Record 9 is composed of Subrecords 2, 3, and 5. If IENGIN is 4, Record 9 is composed of Subrecords 2, 4, and 6. If IENGIN is 5, Record 9 is composed of Subrecords 2, 5, and 6. (See Table H.1 for the description of the subrecords of Record 9.)

Arguments:

SUBREC, VARCODE, IALL, ICTRL

where

SUBREC)
VARCOD) are the same as for the other CRECn subroutines
IALL)

ICTRL = an integer variable whose value signals that the value of ICHOOS is to be changed. If ICTRL is 1, the program jumps to the point designated by {Entry point for ICHOOS} in the pseudocode below.

Common blocks:

In addition to the common blocks for the Record 9 variables and flags, CREC9 also contains the common block

COMMON/ENGDAT/NENGIN,IHRK

CREC9 sets the value of IENGIN depending the value of NENGIN and some other variables which are set during the execution of CREC9. The value of ICHOOS is set according to the value of IHRK and the current value of the program control integer MODE.

Algorithm:

```
Begin
    If non-sequential mode, jump to appropriate part of code
    If NENGIN = 1
       {Subrecord 1}
       Define user-supplied engine data
       Set IENGIN to 1
       RETURN
    End if
    If NENGINE = 2
       {Subrecord 3}
       Choose SHOP5 engine data base
       If non-sequential mode, RETURN
    Else
       {Subrecord 6}
       Define user-supplied engine data base
       If non-sequential mode, RETURN
    Endif
    {Subrecord 2}
    Choose engine configuration per shaft
    {Entry point for ICHOOS}
```

```
    Ask user if SHOP5 must use configuration or if SHOP5 may change
    configuration
    Set value of ICHOOS according to user's answer and current value of
    MODE
    Ask user to choose either SHOP5 gearbox hierarchy or user-supplied
    gearbox hierarchy
    If SHOP5 hierarchy chosen
       {Subrecord 4}
       Ask user to choose SHOP5 hierarchy
    Else
       {Subrecord 5}
       Define user-supplied hierarchy
    End if
    Set value of IENGIN depending on value of NENGIN and user's choice
    gearbox hierarchy
    RETURN
End
```

H.2 SUBROUTINE SHODEF

Purpose:

SHODEF corresponds to the DEFAULT command. It displays the default command of the specified variable.

Common blocks:

```
COMMON/DEVICE/NTTY,NDISK,NDISK2   (see Appendix E.1.)
COMMON/INFO/BEGIN                 (see Appendix H.1.1.)
COMMON/CHARS/WRDLST
COMMON/WRDVAL/WRDLEN,NOWRDS       (see Appendix I.5.)
```

Algorithm:

SHODEF calls the subroutine DICT to find which variable is being specified, then calls the subroutine GETDFT to find the default value of the specified variable. If the variable cannot be found in the default value dictionary, SHODEF prints a message saying that there is no default value for this variable.

H.3 SUBROUTINE SHOW AND ASSOCIATED SUBROUTINES

The SHOW subroutine and its associated subroutines, the SRECn subroutines, correspond to the DISPLAY command. They display the values of variables on the terminal. The n in SRECn is the number of the record which the subroutine displays. The SHOW subroutine controls which of the SRECn subroutines is called.

In program structure, SHOW and the SRECn subroutines resemble the CHANGE subroutine and the CRECn subroutines. Like the CHANGE and CRECn subroutines, the SHOW and SRECn subroutines can execute sequentially or non-sequentially, i.e. the flow of logic can be sequential or it can jump to the appropriate section of code and then exit from the subroutine.

The subroutine SHOW in sequential mode corresponds to the DISPLAY ALL command. The SRECn subroutines in sequential mode correspond to the "DISPLAY record name" command.

In the non-sequential mode, the SHOW and SRECn subroutines work in the same way as the CHANGE and CRECn subroutines.

H.3.1 Subroutine SHOW

Purpose:

This subroutine coordinates the subroutines which display the values of the SHOP5 input variables.

Arguments:

```
IALL
```

where

IALL = an integer which determines whether sequential or non-sequential mode is used
IALL = 1 invokes sequential mode
IALL = 0 invokes non-sequential mode

Common blocks:

```
COMMON/DEVICE/NTTY,NDISK,NDISK2
COMMON/NO2/MODE,INOUT,IPOST,ILPT,ILEGND
COMMON/NO4/IDIMEN,IRESID,IAPPND,IPROP,ISTRUC,IENGIN,ICHOOS,
        IGEN,ICOST
COMMON/NUMBER/NCON,NOPTN,NLIM          (see Appendix E.1.)

COMMON/CHARS/WRDLST
COMMON/WRDVAL/WRDLEN,NOWRDS            (see Appendix I.5.)
COMMON/INFO/BEGIN                      (see Appendix H.1.2.)
```

Algorithm:

```
Begin
    If non-sequential mode is specified, then
       Call DICT to determine which variable to display
       Find which record the variable belongs to
       Use a computed GO TO to jump to the appropriate section of code
    End if
```

```
    {Record 1}
    Call SREC1
    If non-sequential mode, RETURN
    {Record 2}
    Call SREC2
    If non-sequential mode, RETURN
    {Record 3}
    Call SREC3
    If non-sequential mode, RETURN
    {Record 4}
    Call SREC4
    If non-sequential mode, RETURN
    {Record 5}
    Call SREC5
    If non-sequential mode, RETURN
    {Record 6}
    Call SREC6
    If non-sequential mode, RETURN
    {Record 7}
    Call SREC7
    If non-sequential mode, RETURN
    {Record 8}
    Call SREC8
    If non-sequential mode, RETURN
    {Record 9}
    Call SREC9
    If non-sequential mode, RETURN
    {Record 10}
    Call SREC10
    If non-sequential mode, RETURN
    {Record 11}
    Call SREC11
    If non-sequential mode, RETURN
    RETURN
End
```

H.3.2 The SRECn Subroutines

All the SRECn subroutines have the arguments

SUBREC, VARCOD, IALL

where

SUBREC = the character code which identifies the subrecord number (e.g. the character code '02' identifies subrecord 2)
VARCOD = the character code which identifies the variable number
IALL = an integer whose value determines whether sequential or non-sequential mode is used

P48830 PDF [Page: 64 of 80]

All the SRECn subroutines have the common block

COMMON/DEVICE/NTTY,NDISK,NDISK2 (see Appendix E.1.)

The subroutine S3SUB1 (which is called by subroutine SREC3) also uses the common block

COMMON/CHARDM/CHRDIM,NDIM (see Appendix H.1.2)

In addition, each SRECn subroutine contains the common blocks for the variables which the subroutine displays.

The program structure for most of the SRECn subroutines is basically the same as that of the subroutine SHOW. Exceptions are noted below.

H.3.2.1 Subroutine SREC3

This subroutine decides which of two subroutines should be called to display variables in Record 3. The two subroutines are called S3SUB1 and S3SUB2. Subroutine S3SUB1 is used when the program control integer MODE is 0 (indicating search mode in SHOP5); subroutine S3SUB2 is used when MODE is greater than 0 (indicating describe mode in SHOP5).

Both subroutines S3SUB1 and S3SUB2 have the same arguments as the SRECn subroutines (i.e. SUBREC,VARCOD,IALL) and also have the general program structure.

H.3.2.2 Subroutines SREC5 and SREC11

These subroutines have similar structures, so they will be discussed together. They both have the standard SRECn arguments, i.e. SUBREC,VARCOD,IALL. The algorithm they use is as follows:

```
Begin
    If non-sequential mode, determine which variable should be displayed
    If sequential mode, set counter to first variable in the record
    Repeat
       If variable has been re-defined then
          Call GETDES to find description of variable
          Write description and value of variable to screen
       End if
       If sequential mode, go on to next variable
       If non-sequential mode, RETURN
    Until all variables in the record have been processed
    RETURN
End
```

P48830.PDF [Page: 65 of 80]

H.3.2.3 Subroutine SREC9

This subroutine does not follow the sequential/non-sequential structure of the other subroutines because the structure of Record 9 varies, depending on the value of the method control integer IENGIN. (See the description of Record 9 variables in Appendix H.1.2.3.)

SREC9 has the same arguments as the other SRECn subroutines, namely SUBREC, VARCOD, and IALL. Its algorithm is described below:

Algorithm:

```
Begin
    If non-sequential mode then
       If specified variable is not consistent with current value of
       IENGIN, then RETURN
       Jump to appropriate section of code, depending on subrecord number
    End if
    If IENGIN = 1 then
       {Subrecord 1}
       Display data for user-supplied rubber engines
       RETURN
    Else
       {Subrecord 2}
       Display engine configuration per shaft
       If non-sequential mode, RETURN
       If IENGIN = 2 or 3 then
          {Subrecord 3}
          Display year of SHOP5 engine data base
          If non-sequential mode, RETURN
       End if
       If IENGIN = 2 or 4 then
          {Subrecord 4}
          Display SHOP5 gearbox hierarchy
          If non-sequential mode, RETURN
       End if
       If IENGIN = 3 or 5 then
          {Subrecord 5}
          Display user-defined gearbox hierarchy
          If non-sequential mode, RETURN
       End if
       If IENGIN = 4 or 5 then
          {Subrecord 6}
          Display user-defined engine data base
          If non-sequential mode, RETURN
       End if
    End if
End
```

H.4 SUBROUTINE DUMP

Purpose:

This subroutine corresponds to the command DUMP. The subroutine DUMP creates a data file called FIVIN which is used as input for SHOP5. If the user specifies a file other than FIVIN in the DUMP command, the data will be written to this file as well as to FIVIN.

Arguments:

```
        ISTAT
where
        ISTAT  = an integer which returns 0 if the DUMP executed without
                 error, -1 if there was an error
```

Common blocks:

```
        COMMON/DEVICE/NTTY,NDISK,NDISK2          (see Appendix E.1.)
        COMMON/CHARS/WRDLST
        COMMON/WRDVAL/WRDLEN,NOWRDS              (see Appendix I.5.)
        COMMON/LIBFIL/LIBDAT(30,2)
        COMMON/LIBNUM/NFILES                     (see Appendix I.9.)
```

In addition, DUMP contains all the common blocks for the SHOP5 variables and their flags (see Appendix E.1.).

Algorithm:

```
Begin
    Check to see if title and primary input have been defined (using
    subroutine CHKDAT)
    If title or primary input not defined, RETURN
    If file name has not been specified then read a file name
    If file name is not "FIVIN" then
       While (file already exists) or (user does not want to overwrite)
          Read another file name
       End while
    End if
    Write data to FIVIN
    If file name is not "FIVIN" then
       Write data to file
       Add title and filename to library array LIBDAT
    End if
    RETURN
End
```

P48830.PDF [Page: 67 of 80]

H.5 SUBROUTINE ENTER

Purpose:

This subroutine corresponds to the command ENTER. It reads data from an existing file whose name is specified by the user.

Common blocks:

```
COMMON/DEVICE/NTTY,NDISK,NDISK2      (see Appendix E.1.)
COMMON/CHARS/WRDLST
COMMON/WRDVAL/WRDLEN,NOWRDS
COMMON/XIO/LINE
COMMON/XIO2/IJK                      (see Appendix I.5.)

COMMON/LIBFIL/LIBDAT(30,2)
COMMON/LIBNUM/NFILES                 (see Appendix I.9.)
COMMON/CHARDM/CHRDIM,NDIM            (see Appendix H.1.2)
```

In addition, ENTER contains all the common blocks for the SHOP5 variables and their flags (see Appendix E.1.).

Algorithm:

```
Begin
    If filename has not been specified then
       Read filename
    End if
    While (file does not exist) and (user wants another file) do
       Read another filename
    End while
    If user doesn't want another file then
       RETURN
    If file is not in library data then
       Write a message saying file is not listed in library
       Ask user if he wants to enter it anyway
       If user says no, ask for another filename
       If user says yes, set a flag
    End if
    Initialize all flags and variables
    Read in data from file
    If flag is set, enter file name and title into library file
    RETURN
End
```

H.6 SUBROUTINE FIND

Purpose:

FIND corresponds to the command FIND. This subroutine asks the user for a search key, then looks through the library array for files whose title contain the search key. The file names and titles of these files are printed on the terminal.

Common blocks:

```
COMMON/DEVICE/NTTY,NDISK,NDISK2          (see Appendix E.1.)
COMMON/LIBFIL/LIBDAT(30,2)
COMMON/LIBNUM/NFILES                     (see Appendix I.9.)
```

Algorithm:

```
Begin
    Ask user for search key
    if number of files = 0 then
       Write message saying library data doesn't exist
       RETURN
    End if
    If search key is blank then
       Write out all titles and file names in the library file
    Else
       Write out titles and file names for files where the title
       contains the search key
    End if
    RETURN
End
```

H.7 SUBROUTINE HELP

Purpose:

HELP corresponds to the command HELP. This subroutine lists the commands available in FIVPRE, or it lists information on a specified FIVPRE command, depending on the argument INDEX.

Arguments:

```
INDEX
```

where

INDEX = an integer which determines which command is to be explained (see algorithm)

```
Algorithm:

Begin
    If INDEX = 0 then
       Write the list of FIVPRE commands
    Else if INDEX = 1 then
       Write information about command CHANGE
    Else if INDEX = 2 then
       Write information about command DEFAULT
    Else if INDEX = 3 then
       Write information about command DISPLAY
    Else if INDEX = 4 then
       Write information about command ENTER
    Else if INDEX = 5 then
       Write information about command EXIT
    Else if INDEX = 6 then
       Write information about command DUMP
    Else if INDEX = 7 then
       Write information about command FIND
    Else if INDEX = 8 then
       Write information about command HELP
    Else if INDEX = 9 then
       Write information about command NEW
    Else if INDEX = 10 then
       Write an explanation of the special input symbols
    Else if INDEX = -2 then
       Write a message saying that command is not uniquely specified
       Write the list of FIVPRE commands
    Else if INDEX = -1 then
       Write a message saying that no information is available for that
       topic
       Write the list of FIVPRE commands
    End if
    RETURN
End
```

P48830.PDF [Page: 70 of 80]

TABLE H.1: SUBRECORDS OF RECORD 9

Subrecord Number	Description	Variables in the Subrecord
1	User-supplied engine data	SPW2, SFCR
2	Engine configuration per shaft	ITYPE
3	SHOP5 engine data base	IFUTUR
4	SHOP5 hierarchy	IHIAR
5	User hierarchy	HIARK(I),I=1,6
6	User engine data base	SPW2B, NTURB, NDIES, NAMTUR(I), NAMDIE(I), POWT(I), POWD(I), SFCT(I), SFCD(I), WTT(I), WTD(I)

APPENDIX I

UTILITY SUBROUTINES

I.1 SUBROUTINE CAP

Purpose:

This subroutine capitalizes all lower-case letters in a character string.

Arguments:

```
LINE
```

where

LINE = the character string to be capitalized

Algorithm:

```
Begin
    Calculate the difference between the ASCII decimal equivalent (ADE) of
    'A' and the ADE of 'a' using FORTRAN function ICHAR
    For I = 1 to (length of LINE) do
       If I'th character of LINE is a lower-case letter, then add on DIFF
       (the difference calculated above) to the ADE of the character
    End for
    RETURN
End
```

I.2 SUBROUTINE STRIP

Purpose:

This subroutine strips off the leading blanks in a character string and also returns the position of the last non-blank character in the string.

Arguments:

```
LINE, IBLANK
```

where

LINE = character string to be processed
IBLANK = the position of the last non-blank character in LINE

Algorithm:

```
Begin
    Copy characters from LINE into STRING (another character string)
    starting at first non-blank character in LINE
    Starting at end of STRING and moving toward beginning of STRING, look
    at each character of STRING until a non-blank character is found;
    record its position in IBLANK
    Copy STRING into LINE
    RETURN
End
```

I.3 LOGICAL FUNCTION NUMBER

Purpose:

This function checks to see if a character string is a number. It returns true if the character string is a number, and false if the character string is not a number, i.e. if the string contains any non-numeric characters other than '+', '-', or '.', or if the string contains any imbedded blanks. A number may not contain more than one sign ('+' or '-') or more than one decimal point ('.').

Arguments:

LINE, ISTART, IEND

where

LINE = character string to be checked
ISTART = the starting position in LINE
IEND = the last position in LINE to be processed

Algorithm:

```
Begin
    Find first character in LINE which is not a '+','-',or a blank
    For I = (current character position) to IEND do
       If I'th character in LINE is not a numeric character or it is the
       second occurrence of a decimal point, then NUMBER is false
    End
    RETURN
End
```

I.4 SUBROUTINE CONVER

Purpose:

This subroutine converts a character string into a real number (e.g. '124' would be converted to the real number 124).

Arguments:

STR, ANSWER

where

STR = character string to be converted
ANSWER = the real number which is the result of the conversion

Algorithm:

```
Begin
    Right-justify the characters in STR
    Use the FORTRAN internal READ to convert STR into a number
    RETURN
End
```

I.5 SUBROUTINE PLINE

Purpose:

This subroutine takes a character string and breaks it up into a word list. Words are delimited by blanks, '=', semicolons, or commas. Semicolons, commas, and '=' are also considered to be words.

Arguments:

TYPO

where

TYPO = an integer which determines whether a line of input is read from the terminal

Common blocks:

```
COMMON/DEVICE/NTTY,NDISK,NDISK2          (see Appendix E.1.)
COMMON/CHARS/WRDLST
COMMON/WRDVAL/WRDLEN,NOWRDS
COMMON/XIO/LINE
COMMON/XIO2/IJK
```

where

```
        WRDLST = the word list (an array of character strings) which
                 results from the processing of LINE
        WRDLEN = an integer array which contains the length of the words
                 in WRDLST
        NOWRDS = the number of words in WRDLST
        LINE   = the character string to be processed
        IJK    = the last non-blank character in LINE
```

Algorithm:

```
Begin
    If TYPO = 1, read in a line of input (LINE) from the terminal
    Call CAP to capitalize the line
    Call STRIP to strip off leading blanks
    For I = 1 to (end of character string) do
       TEMP = I'th character in LINE
       If TEMP is a blank, then
          Finish processing the current word
       Else if TEMP is a '=', semicolon, or comma, then
          Finish processing the current word
          Add TEMP to the word list
       Else
          Add the character to the current word
       End if
    End for
    Scan through word list and remove all words with length = 0
    RETURN
End
```

I.6 SUBROUTINE READIN

Purpose:

This subroutine reads in a value from the terminal and checks to see if the value is of the specified type (integer, real, or character). If the value is not the right type (e.g. character when it should be real), the subroutine prompts the user until an acceptable value is entered. READIN checks for the "*" input before processing the value. It can also check for the word "DEF".

Arguments:

DTYPE, IVAL, RVAL, CVAL, PROMPT, IDEFLT

where

DTYPE = a character variable which specifies which data type to read in. DTYPE can have the values 'I', 'R', or 'C' to specify integer, real, or character variables.
IVAL = the returned integer value
RVAL = the returned real value
CVAL = the returned character value
PROMPT = a character variable containing the phrase used to prompt for input
IDEFLT = an integer whose value determines whether the word "DEF" should be recognized:
If READIN is called with IDEFLT = 1, then READIN will check for the word "DEF". If the word "DEF" is found, the variable IDEFLT has the value 99 on return to the main program.

Note that only one of the arguments IVAL, RVAL, or CVAL is used to return the value which is read in. IVAL is used when DTYPE = 'I', RVAL is used when DTYPE = 'R', and CVAL is used when DTYPE = 'C'. Dummy arguments should be used for the two arguments which do not return the value. For example, suppose the value to be read in is an integer. READIN might be called with the statement

```
CALL READIN('I',INUM,DUMMY,DUMCHR,'ENTER AN INTEGER',0)
```

The reason for these multiple arguments is that FORTRAN does not convert data types when values are passed through a subroutine's arguments. If an argument is declared to be real in the subroutine, but the calling statement uses an integer variable for that argument, then the real value returned by the subroutine is not converted to an integer value.

Common blocks:

```
COMMON/DEVICE/NTTY,NDISK,NDISK2
```
(see Appendix E.1.)

Algorithm:

```
Begin
    Repeat
       Write PROMPT
       Read a character string from the terminal
       Call CAP to capitalize the string
       Call STRIP to strip leading blanks
```

```
      If the first character is a '*', then
         RETURN
      Endif
      If data type is numeric and string is not a number then
         If IDEFLT = 1 and string = 'DEF' then
            Set value of IDEFLT
            RETURN
         Else
            Write an error message
         End if
      End if
   Until user has entered acceptable input
   If data type is integer or real and string is a number then
      Convert character string to a number
   End if
   If data type is character, CVAL = string
   RETURN
End
```

I.7 SUBROUTINE INITIA

Purpose:

INITIA initializes the values of all the SHOP5 variables and their flags. The variables are initialized to 0, and the flags are initialized to false. Character variables are initialized to blanks.

Common blocks:

INITIA contains all the common blocks for the SHOP5 variables and their flags (see Appendix E.1.).

I.8 SUBROUTINE CHKDAT

Purpose:

This subroutine checks to see if the title (Record 1) and the primary input (Record 3) have been defined. These two records are required data and do not have default values.

Arguments:

IERR

where

IERR = an integer whose value is returned as 0 if both Record 1 and Record 3 are defined, -1 if Record 1 is undefined, and -2 if Record 3 is undefined or incompletely defined.

I.9 SUBROUTINE LIBINT

Purpose:

This subroutine updates the library file, FIVLIB, by checking to see that all the files listed in FIVLIB actually exist. If a file listed in FIVLIB does not exist on the system, LIBINT deletes that entry from FIVLIB.

Common blocks:

```
COMMON/DEVICE/NTTY,NDISK,NDISK2     (see Appendix E.1.)
COMMON/LIBFIL/LIBDAT(30,2)
COMMON/LIBNUM/NFILES
```

where

```
LIBDAT = an array of character strings which contains the titles
         and file names of files created by FIVPRE LIBDAT(I,1) is
         the list of titles LIBDAT(I,2) is the list of file names
NFILES = the number of files listed in LIBDAT
```

Algorithm:

```
Begin
    Read data from FIVLIB into the array LIBDAT
    For each entry in LIBDAT do
       Try to open file using FORTRAN OPEN statement with STATUS = 'OLD'
       If there is an error, then delete the file name and title from
       LIBDAT (i.e. set file name and title to blanks)
    End for
    Remove blank entries from LIBDAT
    RETURN
End
```

REFERENCES

1. Colwell, J.L.: "SHOP5: A Frigate/Destroyer Exploration Model - User's Manual", DREA Technical Memorandum in Review

2. Hally, David and C. Ann Dent: "GETWRD Package", DREA Technical Memorandum 84/D, March 1984.

UNLIMITED DISTRIBUTION

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1 ORIGINATING ACTIVITY DREA	2a. DOCUMENT SECURITY CLASSIFICATION UNCLASSIFIED 2b GROUP
3 DOCUMENT TITLE FIVPRE: A PRE-PROCESSOR FOR THE CONCEPT EXPLORATION MODEL SHOP5	
4 DESCRIPTIVE NOTES (Type of report and inclusive dates) Technical Communication	
5 AUTHOR(S) (Last name, first name, middle initial) Lee, Bonny; Colwell, J.L., Godreau, P.V.	
6 DOCUMENT DATE AUGUST 1985	7a TOTAL NO OF PAGES 78 / 7b NO. OF REFS 2
8a. PROJECT OR GRANT NO.	9a ORIGINATOR'S DOCUMENT NUMBER(S) DREA TECHNICAL COMMUNICATION 85/310
8b CONTRACT NO.	9b. OTHER DOCUMENT NO.(S) (Any other numbers that may be assigned this document)
10 DISTRIBUTION STATEMENT	
11 SUPPLEMENTARY NOTES	12. SPONSORING ACTIVITY DREA

13 ABSTRACT

FIVPRE is an interactive pre-processor for SHOP5, the DREA Concept Exploration Model for conventional monohull frigates and destroyers. FIVPRE provides the user with a simple means to create and modify the input files for SHOP5. Prior knowledge of the format of these files is not required. Through a simple command language, FIVPRE can be used to define new or change existing values of SHOP5 input parameters from the terminal. FIVPRE also keeps records of the files it has created. This Technical Communication is a user's manual for FIVPRE, describing the FIVPRE commands and giving examples of terminal sessions. The appendices contain a brief description of SHOP5 input and program documentation of FIVPRE, including descriptions of all the procedures in FIVPRE.

DSIS
79-070

UNCLASSIFIED

Security Classification

KEY WORDS

Pre-processor
SHOP5
FIVPRE
Naval Architecture
Frigate
Destroyer
Concept Exploration Model

C.5-04076
58830

INSTRUCTIONS

1 ORIGINATING ACTIVITY Enter the name and address of the organization issuing the document.

2a DOCUMENT SECURITY CLASSIFICATION Enter the overall security classification of the document including special warning terms whenever applicable

2b GROUP Enter security reclassification group number. The three groups are defined in Appendix 'M' of the DRB Security Regulations

3 DOCUMENT TITLE Enter the complete document title in all capital letters Titles in all cases should be unclassified If a sufficiently descriptive title cannot be selected without classification, show title classification with the usual one-capital-letter abbreviation in parentheses immediately following the title.

4 DESCRIPTIVE NOTES Enter the category of document, e.g. technical report, technical note or technical letter. If appropriate, enter the type of document, e g interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.

5 AUTHOR(S) Enter the name(s) of author(s) as shown on or in the document. Enter last name, first name, middle initial. If military, show rank. The name of the principal author is an absolute minimum requirement.

6 DOCUMENT DATE Enter the date (month, year) of Establishment approval for publication of the document.

7a TOTAL NUMBER OF PAGES The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information

7b NUMBER OF REFERENCES Enter the total number of references cited in the document.

8a PROJECT OR GRANT NUMBER If appropriate, enter the applicable research and development project or grant number under which the document was written.

8b CONTRACT NUMBER If appropriate, enter the applicable number under which the document was written.

9a ORIGINATOR'S DOCUMENT NUMBER(S) Enter the official document number by which the document will be identified and controlled by the originating activity This number must be unique to this document

9b. OTHER DOCUMENT NUMBER(S) If the document has been assigned any other document numbers (either by the originator or by the sponsor), also enter this number(s).

10. DISTRIBUTION STATEMENT Enter any limitations on further dissemination of the document, other than those imposed by security classification, using standard statements such as

(1) "Qualified requesters may obtain copies of this document from their defence documentation center."

(2) "Announcement and dissemination of this document is not authorized without prior approval from originating activity."

11. SUPPLEMENTARY NOTES Use for additional explanatory notes

12. SPONSORING ACTIVITY Enter the name of the departmental project office or laboratory sponsoring the research and development. Include address.

13. ABSTRACT Enter an abstract giving a brief and factual summary of the document, even though it may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall end with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (TS), (S), (C), (R), or (U)

The length of the abstract should be limited to 20 single-spaced standard typewritten lines, 7½ inches long

14 KEY WORDS Key words are technically meaningful terms or short phrases that characterize a document and could be helpful in cataloging the document. Key words should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context.